'68' MICRO JOURNAL

$2.95 USA

Australia	A $ 4.75	New Zealand	NZ $ 4.75
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30:-SEK

VOLUME IV ISSUE X • Devoted to the 68XX User • October 1982

"Small Computers Doing Big Things"

YOUR CHOICE-smart either way

- Over 140 software driven functions
- 82 x 24 or 82 x 20 screen format — software selectable
- High resolution 7 x 12 matrix characters — P-31 green phosphor
- Upper/lower case character set — plus graphics character set
- 56-key alphanumeric keyboard — plus 12-key cursor, numeric pad
- Internal editing functions — insert, delete, scroll, roll, slide, etc.
- Parallel printer I/O port
- 50 to 38,400 baud operation — programmable
- Cursor type, cursor position, print control characters, protected fields, shift inversion, dual intensity and many other features

8212 — twelve-inch diagnonal screen or **8209** — nine-inch diagnonal screen

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Pascal for 6809

Pascal for the 6809 is a true native code compiler. Unlike the usual P-code Pascals which run in an interpretive manner, ours produces efficient assembly language mnemonics which can be assembled and run directly. This compiler is available for both 6809 FLEX™ and UniFLEX™. Many features not found in other Pascal systems were implemented while avoiding those features completely non-standard. Features of the Pascal system include:

- Supports most of Jensen and Wirth specification
- Produces fast and efficient 6809, native code
- FLEX run-time package may be trimmed
- Double precision real numbers (16.8 digits)
- Implements scalar, subrange and structured data types
- Standard I/0 using file buffer pointers
- Dynamic storage allocation
- Ability to call other Pascal programs
- FLEX version may call assembly language programs
- Buffered or single character terminal input
- Standard math functions: SIN, COS, ARCTAN, EXP, LN, SQR, SQRT
- Random number generator function
- Many usable, sample programs included
- UniFLEX version supports:
  - Random file positioning
  - Ability to call various UniFLEX system routines
  - Ability to execute UniFLEX utility commands

Pascal on diskette for 5" and 8" 6809 FLEX is available for $200.00 The 5" version requires two disk drives. The UniFLEX version is $300.00 and includes one year of maintenance. All orders should include 3 percent for postage and handling (10 percent on foreign orders).

™FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.

technical systems consultants, inc.

111 Providence Rd., Chapel Hill, N.C. 27514

(919) 493-1451

'68' MICRO JOURNAL

Portions of text prepared using the following.

SWTPC 6800-6809-OMAF2-CDS1-CT82-Sprint 3
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216

EDITOR - WORD PROCESSOR
Technical Systems Consultants, Inc.
Box 2573, W. Lafayette, IN 47906
FLEX is TM of TSC

GIMIX Super Mainframe-Assorted memory boards
GIMIX Inc.
1337 West 37th Place
Chicago, Il 60609

CONTENTS

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68 Micro Journal is published 12 times a year by Computer Publishing, Inc. Second class postage paid at Hixson, Tennessee and additional entries. Postmaster: send Form 3579 to 68 Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES
USA
1-Year $24.50 2-Years $42.50 3-Years $64.50
FOREIGN
See Page 52

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors full name, address, date and telephone number. It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands ONLY should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

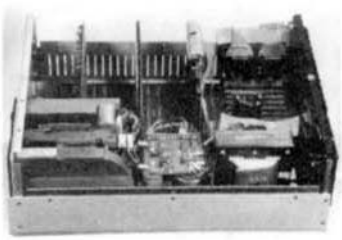
Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

FLEX - OS-9 LEVEL ONE - UNIFLEX - OS-9 LEVEL TWO

ONLY GIMIX Systems can be configured to run any of these.

GIMIX systems utilize the most powerful 6809 operating systems: FLEX, UniFLEX, OS-9 LEVEL ONE and TWO -- the systems the PROs use. This means a wide selection of software to choose from as well the ability to develop sophisticated, multi-user/multi-tasking programs on your **GIMIX** System.



The GIMIX CLASSY CHASSIS™ consists of a heavy-weight aluminum mainframe cabinet which provides more than ample protection for the electronics and 1 or 2 optional 5¼" drives.

Backpanel connectors can be added for convenient connection of terminals, printers, drives and other peripherals.

A 3 position locking keyswitch enables users to disable the front panel reset button to prevent accidental or unauthorized tampering with the system.

The **GIMIX** system mother board provides fifteen 50 pin slots and eight 30 pin I/O slots -- the most room for expansion of any SS50 system available. The on board baud rate generator features 11 standard baud rates, 75 to 38.4K, for maximum versatility and compatibility with other systems. Extended address decoding allows the I/O block to be addressed anywhere in the 1 megabyte address space. All components feature Goid plated connectors for a lifetime of solid connections. All boards are fully buffered for maximum system expansion.

Each **GIMIX** Mainframe System is equipped with an industrial quality power supply featuring a ferro-resonant constant voltage transformer to insure against problems caused by adverse power input conditions such as A.C. line voltage fluctuations etc. The supply provides 8 volts at 30 amps and plus or minus 16 volts at 5 amps, more than enough capacity to power a fully loaded system and two internal drives.

The 2MHz GIMIX 6809 PLUS CPU board includes a time of day clock with battery back-up and 6840 programmable timer to provide the programmer with convenient, accurate time reference. Later addition of 9511 or 9512 arithmetic processors is provided for on the board. The unique **GIMIX** design enables software selection of either OS-9 or FLEX, both included in many complete **GIMIX** systems.

GIMIX STATIC RAM boards require no complicated refresh timing cycles or clocks for data retention. GIMIX memory boards are guaranteed for 2 MHz operation with no wait state or clock stretching required.

Our low power NMOS RAM requires less than 3/4 amp at 8V for a fully populated 64K board. For critical situations, our non-volatile 64K byte CMOS static RAM boards with built in battery back-up retain data even with system power removed. A fully charged battery will power this board for a minimum of 21 days. A write protect switch permits CMOS boards to be used for PROM/ROM emulation and software debugging.

The GIMIX DMA controller leaves the processor free to perform other tasks during disk transfers - an important feature for multi-user/multi-tasking systems where processor time allocation is critical. The DMA board will accomodate up to 4 drives 5¼" or 8" in any combination running single or double density single or double headed. Programmed I/O Disk Controllers are also available.

GIMIX systems are designed with ultimate **RELIABILITY** in mind. You can choose from the below featured systems or select from our wide variety of components to build a custom package to suit your needs.

GIMIX 2MHz 6809 System including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 56KB STATIC RAM, 2 SERIAL PORTS W/CABLES, GMXBUG MONITOR, FLEX, and OS-9 LEVEL 1 . . . . . **$3248.49**
FOR TWO 5¼" 40 TRACK DSDD DRIVES ADD . . . . . **$ 900.00**
GIMIX 128KB WINCHESTER SYSTEM including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 128KB STATIC RAM, 4 SERIAL PORTS W/CABLES, 5¼" 80 TRACK DSDD FLOPPY DISK DRIVE, 19MB 5¼" WINCHESTER HARD DISK, OS9 LEVEL 2, EDITOR AND ASSEMBLER **$8998.09**

50HZ Versions Available, 8" Drives Available — Contact GIMIX for Prices and Information.

The Sun Never Sets On A GIMIX!

GIMIX users are found on every continent, including Antarctica. A representative group of GIMIX users includes: **Government Research and Scientific Organizations** in Australia, Canada, U.K. and in the U.S.; NASA, Oak Ridge, White Plains, Fermilab, Argonne, Scripps, Sloan Kettering, Los Alamos National Labs, AURA. **Universities:** Carleton, Waterloo, Royal Military College, in Canada; Trier in Germany; and in the U.S.; Stanford, SUNY, Harvard, UCSD, Mississippi, Georgia Tech. **Industrial users** in Hong Kong, Malaysia, South Africa, Germany, Sweden, and in the U.S.; GTE, Becton Dickinson, American Hoechst, Monsanto, Allied, Honeywell, Perkin Elmer, Johnson Controls, Associated Press, Aydin, Newkirk Electric, Revere Sugar, HI-G/AMS Controls, Chevron. **Computer mainframe and peripheral manufacturers,** IBM, OKI, Computer Peripherals Inc., Qume, Floating Point Systems. **Software houses;** Microware, T.S.C., Lucidata, Norpak, Talbot, Stylo Systems, AAA, HHH, Frank Hogg Labs, Epstein Associates, Softwest, Dynasoft, Research Resources U.K., Microworks, Meta Lab, Computerized Business Systems.

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

GIMIX INC.

The Company that delivers

Quality Electronic products since 1975.

Build performance into your system

with OS-9** software tools

Unix*-based, multitasking, modular, and versatile: these key features are some of the reasons why more 6809 computer manufacturers have selected OS-9 as their standard operating system than any other. And OS-9 has been put to work by thousands of users in almost every conceivable computer application in business, science, industry, education, and government.

Your operating system should not be a barrier between you and your computer. OS-9 is very friendly and easy to use. Its modular structure makes it easy to customize, plus its comprehensive documentation shows you exactly how to interface it to just about any I/O device.

OS-9's advanced features unleash the performance potential of almost any 6809 computer — large or small. In many respects the OS-9/6809 combination is more powerful than many *minicomputers!*

There are two basic versions of OS-9. Both have the same basic features and capabilities. OS-9 Level One runs on small to medium sized systems having up to 64K memory. The Level Two version runs on medium to large size systems having memory management hardware and up to 1 megabyte of memory, and includes record and file locking for multiuser database applications.

Here are just a few reasons why you should insist on OS-9 for your microcomputer system.

- Over 40 utility commands
- Friendly "Shell" command interpreter
- Tree-structured multilevel file directories
- Full timesharing support with log-in and file security
- Fast, secure random and sequential access files
- Comprehensive English language error messages
- Compact real-time multitasking executive
- Hardware or software memory management
- Device independent interrupt-driven I/O
- Fully ROMable for small control systems
- Standard versions available from manufacturers of most popular 6809 computers

OS-9 PASCAL Language Compiler

- most complete and versatile PASCAL available for the 6809
- capable of generating P-code for interpretive execution while debugging OR
- highly optimized 6809 assembly language source code output for maximum speed
- "virtual memory" P-code interpreter lets you run large PASCAL programs

CIS COBOL * Compiler**

- ideal for most demanding business applications
- features ISAM, Debug, ACCEPT/DISPLAY and Interprogram Communications modules
- retains full compatibility with CP/M software
- meets ANSI 1974 Level One COBOL standard and is GSA certified
- Also available-FORMS 2 automatic program generator for easy interative design of screen oriented applications.

BASIC09 Structured Basic Interactive Compiler**

- fastest and most comprehensive full Basic language available for the 6809
- combines standard Basic with the best features of PASCAL
- features compiler speed, interpreter friendliness and superlative debugging facilities
- o tion available: Run B.. a ROMable run-time system for compiled Basic 09

C Language Compiler

- complete implementation of the UNIX version 7 C language
- includes INT, CHAR, SIGNED, UNSIGNED, FLOAT AND LONG data types, structures, unions, standard C library and a full preprocessor with macro definitions
- generates fully reentrant 6809 assembly language source code output

For more information contact your computer supplier, or

MICROWARE

Microware Systems Corporation
5835 Grand Avenue, Des Moines, Iowa 50312 515-279-8844 • Telex 910-520-2535

*Unix is a trademark of Bell Laboratories. ***CIS Cobol is a trademark of Micro Focus, Inc. OS-9** and Basic09 are trademarks of Microware and Motorola, Inc.

LEVEL TWO

Expand your 6809 computer to a fast, efficient multi-user system utilizing up to one megabyte of memory, almost any I/O device, and comprehensive implementations of the most-wanted programming languages: Basic09**, C, Pascal, Cobol, and Assembler.

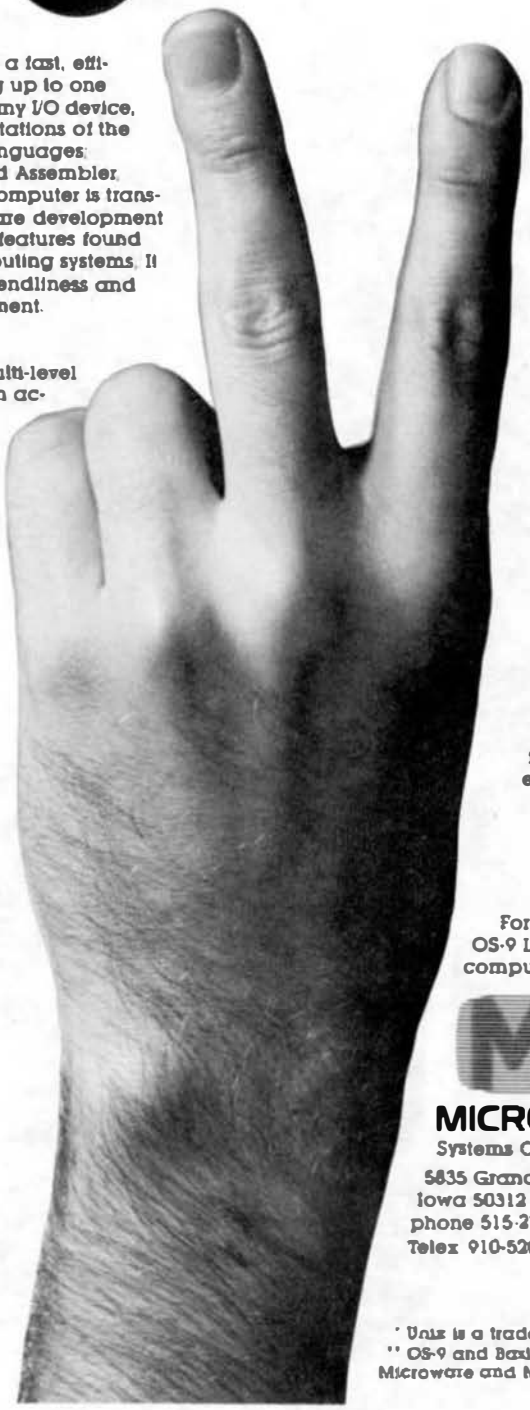
With OS-9** Level Two, your computer is transformed into the ultimate software development system with performance and features found only on large and costly computing systems. It brings to your fingertips the friendliness and power of a Unix*-style environment.

As a multi-user system...

OS-9 Level Two excels with a multi-level directory system, a fast random access file system with record lockout, user name/password logon protection, "pipes" for interprogram communication, and full file security. The versatile "Shell" command interpreter makes it easy for each user to run interactive or multiple background programs with I/O redirection to or from any file or I/O device.

As a real-time system...

Your OS-9 based computer can sense, monitor, control, and communicate with the real world thanks to OS-9's highly modular and user expandable structure. Adding customized I/O to OS-9 is almost too easy; software interfaces are simple, modular, and well documented. That is why OS-9 users routinely interface it to almost every kind of peripheral device and instrumentation.



For large systems...

OS-9 Level Two can handle over one megabyte of RAM and ROM memory with extraordinary efficiency. When, as is often the case, two or more tasks run the same program (such as Basic09) they automatically "share" just one copy in memory. Also, OS-9 Level Two typically resides in less than 24K memory. Savings like these give OS-9 based systems large capacity without having to resort to performance-robbing techniques such as disk swapping.

OS-9 Level Two is available exclusively from manufacturers of most popular 6809 computers equipped with memory management hardware. They offer versions specifically tailored to their computers for use with both new and existing systems.

For more information about OS-9 Level Two contact your computer supplier, or

MICROWARE
Systems Corporation
5835 Grand Avenue, Des Moines,
Iowa 50312
phone 515-279-8844
Telex 910-520-2535

* Unix is a trademark of Bell Laboratories.
** OS-9 and Basic09 are trademarks of Microware and Motorola, Inc.

C compilers and C ross compilers

TELECON'S C COMPILERS OFFER YOU

- FULL C
- UNIX* Ver. 7 COMPATABILITY
- NO ROYALTIES ON GENERATED CODE
- GENERATED CODE IS REENTRANT
- C AND ASSEMBLY SOURCE MAY BE INTERMIXED
- UPGRADES & SUPPORT FOR 1 YEAR

IN THESE CONFIGURATIONS:

HOST	6809 TARGET	PDP-11*/LSI-11* TARGET	8080/(Z80) TARGET	8088/8086 TARGET
FLEX*/UNIFLEX* **OS-9***	$200.00 WITHOUT FLOAT $350.00 WITH FLOAT	500.00	500.00	500.00
RT-11*/RSX-11* PDP-11*	500.00	200.00 WITHOUT FLOAT 350.00 WITH FLOAT	500.00	500.00
CP/M* 8080/(Z80)	500.00	500.00	200.00 WITHOUT FLOAT 350.00 WITH FLOAT	500.00
PCDOS*/MSDOS* 8088/8086	500.00	500.00	500.00	200.00 WITHOUT FLOAT 350.00 WITH FLOAT

OS-9 Version Available Now

Others Pending

C SOURCE AVAILABLE FOR $2,500^{00}$

SO . . . IF YOU'RE READY TO MOVE UP TO C ...

CALL

408-275-1659

TELECON SYSTEMS
90 E. Gish Road, Suite 25
San Jose, CA 95112

*PCDOS is a trademark of IBM CORP. MSDOS is a trademark of MICROSOFT. UNIX is a trademark of BELL LABS. RT-11/RSX-11/PDP-11 is a trademark of Digital Equipment Corporation. FLEX/UNIFLEX is a trademark of Technical Systems consultants. CP/M is a trademark of Digital Research. OS-9 is a trademark of Microware & Motorola.

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

An Editorial

I don't often label an opinion as directly as this one, but this is pure opinion, and you can take it or leave it. I think the Microprocessor industry is suffering from the "Bigger and/or Faster is Better" syndrome. The Microprocessor is a natural solution to many control and instrumentation applications. In many of those, cost is a primary consideration. Many manufacturers of board assemblies are phasing out the 1 MHZ processors in favor of 2 MHZ versions. Today, I checked prices of the 6821 and the 68B21 PIA. The B version is the 2 MHZ. The best price I could find was more than two times the price for the plain 6821. Fast memories are more expensive. The processor itself costs considerably more. Yet, small quantity manufacturers who depend on the suppliers of boards for their product, are facing a choice of paying much more for components or building their own boards. Something like a CPU board, if you have never checked, costs in the neighborhood of $3000 to design. That is after you have a schematic diagram done. That cost is for the Artwork for a typical double sided board of the approximate size of a SS-50 board. Memory boards are a bit more "regular" in pattern and don't cost quite as much to have done.

Then, we face the oncoming 68000, which I'm sure, Motorola hopes will obsolete all the 6800 and 6809 applications. Don't get me wrong, the 68000 is great if you need its speed and power, but you pay for it.

I have a 6809 in an instrument that takes readings and updates the display about every 1.5 seconds. The reading interval is largely set by the time the processor takes to process the information for display. Have you ever tried to read a digital display that is changing, and updating 10 times a second? It is impossible. If I had a faster processor, I would have to put a delay loop in the program so the display refresh time would be about what I have now. That sort of refresh time is just about long enough to allow the user to read the numbers before they change. Perhaps a display quickly changing back and forth between 100 and 101 can be read even with a high update rate, but try reading a display that is changing from 99 to 100 rapidly! It is most likely that you will read 199 as the value. You will have to agree that it would be rather silly to pay for a faster processor or one with more power, and then put a delay loop in the program. I am afraid that we will have no choice in the future. I sincerely hope that the manufacturers of both hardware and software will continue to keep the lower performance versions available for applications where they are "good enough" and more economical.

I was recently disappointed to learn that SWTPC has discontinued the 1 MHZ version of the MP-09 processor board at $195 and now sells only the 2 MHZ version at $295. Now I can pay $100 more for the processor only to change the crystal or upgrade all the other system components to 2 MHZ versions as well. I have little alternative but to advise the company for which I consult to build its own processor board for the application.

OK, I'll stop. I hope you (and some of the hardware suppliers) see what I am getting at.

PERFECT NUMBERS

A few days ago, I received a letter from Keith Alexander, containing a program to find "Perfect Numbers". A perfect number is a number, the sum of whose factors adds up to the original number. The factors are not just the "prime factors" but all the factor pairs that are found, and of course 1, since all numbers are divisible by 1. The first perfect number is 6 (3*2*1). The second is 28 (2*14, 4*7). I sent a note back to Keith after looking the program over a bit. He had found factors by testing all the numbers from 1 to half of the number being tested, and testing the quotient to see if it was integer. Keith had translated the program between several languages, and had ended up using REAL variables in the pascal version. I had decided to use integers, which speeded up the program considerably. A division is even in Pascal if TEST NUMB MOD TEST FACTOR = 0. That worked fine for finding factors. Keith had used test factors from 1 to half the test number, and had added the test factor to SUM, which was initialized to 1. I noted that each test factor that resulted in an even division actually found two factors, and added both test factor and quotient to the sum. It then was an extension of my old prime number programs, to test only numbers up to the square root of the number being tested.

Keith reported that his program had found the first three perfect numbers in several minutes, but the fourth was not found in 10 hours. My speeded up version found the fourth number in about 10 minutes. I decided to run the program overnight and found after 10 hours, that the fifth perfect number had not been found, though I had reached the limit of 30000 for the number being tested. Since the integer arithmetic limit is 32767, I had stopped there. Next I wrote the program in Dugger's C using long integers, and ran it overnight to a test number of about 100000, with no further perfects found. I decided to list the factors of the perfect numbers I had found, and I saw a pattern to all of them. The listing is included here, and it will test numbers to over 8,000,000, though I don't recommend running it to that limit, since I calculate that it would take several years

number	factors	second factor	short of a power of 2
6	2	3	1
28	2	14	2
	4	7	1
496	2	248	8
	4	124	4
	8	62	2
	16	31	1
8128	2	4064	32
	4	2032	16
	8	1016	8
	16	508	4
	32	254	2
	64	127	1

These numbers are all of the form 2(^)(2*n+1) - 2(^)n. Perfect powers of 2 might be perfect numbers, but they always have other factors. What seems to make these perfect is that the "last second factor" (in the tables above) is always one less than a power of 2, and it is always prime. (In terms of our description of the number, that factor is 2(^)(n+1) -1. Seeing the pattern, I wrote a program in TSC Extended BASIC using floating point arithmetic, which is 64 bits, and will give complete integer results to about 18 digits. I set up a loop for n = 1 to 31 to see if the 2(^)(n+1) - 1 values were prime. If a prime was found, I calculated the "perfect number" from the formula above.

This program produced 7 numbers. It doesn't find 6, the first perfect, but I could fix it to do so. The first new Perfect that popped out, was 33,550,336. I wrote a program in BASIC to find its factors and verified that it is a perfect number. The Extended BASIC program found 6 of its 7 numbers in 25 seconds, and the 7th in about

13 minutes. The numbers are

```
                6
               28
              496
             8128
         33550336
       8589869056
     137438691328
2305843008139952130
```

I have since learned that the last number does exceed the range of TSC extended BASIC representation. It actually should be 2305843008139952128, an error of two being present in the last place. The listing of the BASIC program is included here also.

I have no other software that will allow more powers of 2 to be tested. I found that the 2(^)N (built in) function didn't give accurate enough results, and I had to do a subroutine to multiply by 2 repeatedly to arrive at the power of 2 accurately. I found, and point out with pleasure that the TSC floating point package yields exact values as long as the number may be represented within the 63 bits (Plus sign bit) format for floating point numbers.

My Pascal version of Keith's original program is listed here, as is my BASIC program to find the larger perfects. Thanks, Keith, for getting me into an interesting two evening project.

PRICES FALLING

This is a fairly involved story. I was going to devote some column space to remarks about the falling prices of RAM memory. Of course we can all read the ads, and we know that 56K of static RAM may now be purchased for under $400. I have seen the Digital Research Computers board (advertized in '68'). I saw it unpacked, plugged in to an SS-50 computer after removing all the other memory boards, and watched it work perfectly when the power was turned on. When I bought some of my early 8K boards, the going price was around $100 for each 4K chunk of memory. That is about $1600 for 64K.

Now, we've all complained that computers were getting ridiculously expensive, just totally beyond the means of the average hobbyist. Well, maybe a full blown dual 8 inch disk drive system, or one with a hard disk is still there, but there is a system available at a much reduced price. You've all probably seen the ads too. Yup, I'm talking about the TRS-80 Color computer. For around $650, you can have a CC complete with Extended Disk BASIC and 64K of memory. With this, all you have to add is your TV set, and you have a working computer. You can run all the TRS game cartridges, and use a cassette tape player for program storage. Now for an additional $550 to about $950 you can have one or two disk drives, double density, and single or double sided. I opted for the two double sided drives and the $950 price. Now, for around $1600 I have a computer that will run all the FLEX software, two disk drives with storage capacities of 375K each, and a keyboard. Just add one TV set and you have a computer that is very capable.

I'm not implying that it is as nice and as useful as a standard SS-50 bus computer, but it didn't cost as much either. Presently, you must use the CC keyboard and a TV set for running the computer, but I'm told that shortly there will be available both hardware and software solutions to the problem of connecting a standard terminal to the CC. Now for another $650 or so, you can buy a reasonable terminal and have full 80 column by 24 line capabilities. Now, add an Epson MX-80 printer for $450 or so, and you have a "complete" system for less than $3000. I figure that replacement cost for my present SS-50 system would be about $5500. Sure, I know, I have a pair of DSDD 8 inch drives with a total of 2,000,000 bytes storage. I wrote a book last year, and all 345 pages fit on just over half of one of those 8" disks. How many hobbyists are going to write books on their systems. Those 5" DSDD disks wouldn't quite hold my whole book text file but two of them would! Of course the book is not just one big file, it is split into a file for each chapter plus about 50 or so program files. It would be no handicap at all to have to fit it on two disks rather than one.

By the time you read this, I will have written reviews of both the Frank Hogg Laboratory and the Data Comp Division color computer systems. They are both excellent buys. Currently they are quite capable of running lots of FLEX software, and there are add-on's both hardware and software, that will transform these systems into systems with all the capabilities of their big brothers. The best thing about all this is that you don't have to invest all that money at one time. You can start with the computer and have all the CC software to play with. Add the disk drive(s) and one of the FLEX software packages, and you can program in assembler and several compiled languages. Add a terminal and a printer, and you have a useful and capable word processing system. There are several good editor processor combinations available for these systems.

If any of you software or hardware suppliers are listening, here is my list of most needed items for these systems. 1. An inexpensive adaptor for the serial output, to run a parallel printer. Epson wants over $150 for their serial interface. That's a lot to add to a $450 printer. 2. Modem or "download" software to allow communication with another computer via the serial port or the cassette port. 3. That software to allow running a standard terminal with the CC. 4. For those who don't want or can't afford a terminal, a screen editor for the CC. 5. In the way of hardware, some sort of add-on support for user programmable serial and parallel ports. With those, a user may be able to add A/D converters and logic control outputs to run laboratory experiments and collect data on the results. 6. Information on the CC hardware as it is supplied. Addresses of those ports, descriptions of them (standard Motorola 6850, etc.), availability of baud rate signals, specifications of compatible connectors, pinouts, description of printer protocol required by the CC, etc.

Some of you reading this must be CC owners. How about some articles for '68' about the hardware or software you've created to make your CC more useful?

6809 VS 6800

Some of you are still wondering what's so great about the 6809 as compared to the 6800. Those extra instructions don't really look like they are so vastly superior. How about a little example. The 6809 has an instruction PSHS X that pushes the value in X onto the system stack. Try writing a subroutine for the 6800 that does that same thing (a requirement for getting a variable on the stack for "re-entrant" code, ie code that is interruptable). You will find the following to be a subroutine that will do the job of the single 6809 instruction.

```
PSHX  PULA
      STAA STEMP
      PULA
      STAA STEMP+1 SAVE RETURN ADDRESS FOR SUBROUTINE
      STX XTEMP
      LDA XTEMP+1
      PSHA
      LDA XTEMP
      PSHA
      LDA STEMP+1
      PSHA
      LDA STEMP
      PSHA
      RTS
```

All that to do one simple push operation. The main problem with the 6800 is that single stack. To do anything to it in a subroutine, you must first save the return address, modify the stack and then put the return address back before the RTS. In the case of the above, you could replace the instructions starting with LDAA STEMP+1, with LDX #STEMP JMP ,X. Of course then you would have altered the contents of X as well as A. Also, while executing this routine, you are using absolute memory locations to store the contints of X and S. An interrupt during this routine, if the interrupt service routine happens to call this routine again, would be disastrous, because you would write over the XTEMP and STEMP locations, thus losing the data for the call to the subroutine that was interrupted. The only way out would be to use a SEI instruction at the beginning of the PSHX routine and a CLI at the end. Of course with the 6809 the process is a single instruction and no subroutine call is needed.

Want some further examples? How about a routine to add A to X?

```
ADDAX  STX XTEMP
       ADDA XTEMP+1


       STAA XTEMP+1
       LDAA XTEMP
       ADCA #0 TAKE CARE OF CARRY
       STA XTEMP
       LDX XTEMP
       RTS
```

6809 VERSION

```
       LEAX A,X
```

To subtract A from X would be about the same in 6800 code, substituting SUBA and SBCA instructions. In 6809 code, you would use two instructions, NEGA LEAX A,X ! Remember, every time there is a variable in an absolute memory location such as XTEMP, the code is not re-entrant, ie may not be interrupted and called from the interrupt routine. That 6809 code is automatically re-entrant if you use the IRQ, and may be made re-entrant with FIRQ if you save the appropriate registers in the interrupt routine.

It seems that those of us who chose the 68XX at an early date have "lucked out" as the saying goes, largely because Motorola wasn't satisfied with the 6800, but went on to develop the 6809 (and 68000). The beauty of the 6809 instruction set is not immediately apparent until you try to do some operations like the above examples.

Here is Ron's cryptogram. No prizes, but I have a feeling that not very many of you will figure it out.
I will "reveal" the code later.
Should you happen to figure it out,
please write me with your solution.
Your computer won't be much help.

Nonsense as nonsense is
a ball
never to

wonder if words sensible,
lengthy, complex, cryptic
or in "formation"

to appeal to
a
no pattern
cryptographer no, no

Never
in, on, at, or
by an
a, b, c

```
10   REM          FIND PERFECT NUMBERS OF FORM
20   REM          2^(2*N+1) - 2^N
30   REM          FOR THESE TO BE PERFECT, 2^(N+1) - 1
40   REM          MUST BE PRIME.
50   REM          TSC 2^N DOESN'T YIELD PERFECT INTEGER RESULTS
60   REM          SO SUCCESSIVE MULTIPLIES BY 2 WERE USED
70        FOR N = 1 TO 31
80        E=N+1
90        GOSUB 1000
100       TN = N1 - 1
105       IF TN < 4 THEN 160 : REM 1,2,3 ARE PRIMES
130         FOR K=3 TO INT(SQR(TN)) STEP 2
140           IF TN/K = INT (TN/K) THEN 270 : REM NUMBER NOT PRIME
150           NEXT K
160         E= 2*N+1
170         GOSUB 1000
180         NU = N1
200         E=N
210         GOSUB 1000
220         NU = NU - N1
260         PRINT USING '######################',NU
270       NEXT N
280       END
990  REM          SUBROUTINE TO CALCULATE 2^E (RETURNED IN N1)
1000      N1 = 1
1010      FOR I=1 TO E
1020        N1 = N1 * 2
1030      NEXT I
1040      RETURN
```

```
/* PROGRAM TO FIND PERFECT NUMBERS */

$include RLIB.EQU.1

main ()
{
   long t_numb, t_fact, quotient, sum;
   int incr;
   char c;

   printf ("\n\n");
   printf ("starting number for test:");
   scanf ("%ld",&t_numb);
   printf ("\nthe perfect numbers less than 8,000,000 are:\n");

   while (t_numb < 8000000)
   {
       sum = 1;
       t_fact = 2;

/* odd numbers are not divisible by even ones */
       if (t_numb % 2 != 0)
          {
```

```
            incr = 2;           /* even */
            t_fact = 3;
         }
      else
         {
            incr = 1;           /* odd */
            t_fact = 2;
         }

      while (t_fact && t_fact < t_numb)
      {
         if (t_numb % t_fact == 0)
         {
            quotient = t_numb / t_fact;
            sum = sum + t_fact + quotient;
         }
         t_fact = t_fact + incr;
      }
      if (sum == t_numb)
      {
         printf ("%d\n",t_numb);
      }
      if ((c=getkey()) == 27)
         {
            printf ("last number tested %d\n",t_numb);
            exit();
         }
      t_numb = t_numb + 1;
   }
   ring_bell;
}

ring_bell()
{
   int i, j;

   for (i=1; i<10; i++)
   {
      printf("%c",7);
      for (j=1; j<1000; j++);
   }
}

getkey()
{
asm
 LDB $E005
 CLRA
endasm
}
```


COLOR User Notes

Robert L. Nay
5900 Cassandra Smith Rd.
Hixson, Tn. 37343

INTRODUCTION

This issue marks the first anniversary of the "COLOR Users Notes" column; it's hard to believe that it has only been ONE year because of my personal situations. Those of you who have been trying to track me down with letters, or who have been following the address changes, know what I mean. Sometimes the path through life does not follow a very straight line. Just to prove that the 'Almighty Column Writer' is at least somewhat human, this 'average working mans' attempt at keeping food on the table and a roof overhead went something like this:

1. Try to hold a group of people together while the Company you work for is going 'down the tubes' due to the economy. Cost; a couple months pay a many grey hairs.
2. A short stint with Radio Shack as a Computer Service Rep. and Educator; helped set up a new Computer Center in the middle of Atlanta. Their pay scale is not set up to support a 'mature' family and it's needs.
3. Went to work with the '68' Micro Journal staff - for me, a 'dream job'. Again, the demands of the family, and conditions of the economy, do not allow me to stay.
4. Managed to get number one daughter married off; number two daughter graduated from high school and is launching a life of her own: instant EMPTY HOUSE!
5. Now I'm back home again, this time I'll try the 'College Professor' routine (they say if you can't do anything else, you can always teach).

There you have it. A 'normal' year in the 'average working mans' life. Well, the 'Column must go on', so lets get to it!

F-MATE(RS) Version 2.0

To say we are proud of the DATA-COMP Flex Conversion would be the understatement of the century. Steve Odneal has done a superb job with this Software Package. We have yet to turn up a valid 'bug' in the Software, so we are going ahead with the release of Version 2.0. Some significant changes and improvements have been made to include 'convenience features in FLEX, some to correct minor problems, and still others which greatly enhance the usability and function of FLEX and the Color Computer in general. Briefly, the changes are as follows:

1. ESCAPE CONTROL - Most standard FLEX Systems provide for stopping and restarting an output from a FLEX Utility or Pr This feature has been added and applies to both the video and printer outputs. Hitting the defined 'Escape' Key during an output will cause that output to halt at the next <cr>, and provide the b cursor. Hitting the 'Escape' again will continue the output, or hitting a <cr> (the <ENTER> Key) will stop further output and return you to FLEX.
2. PRINTER NOT READY - Two Printer-Not-Ready checks are provided with V2.0. First, the new P.CMD checks to see if the Printer is ready, and if not, allows you to get it ready and go ahead, cancel the print function but go ahead with the output going to the Display, or just return to FLEX. Second, the routine checks the printer status before each character is output, and, should the printer go 'down' after output has begun, you get a warning and the output is diverted to the Screen.

3. RS232 PRINTER CHARACTER BITS - The number of bits output is now user controllable via the User Routines.
4. USER ROUTINES and DATA AREAS - More User Control and Configuration is provided with Version 2.0.
5. DRIVES NOT READY - Code has been added to test the status of the Disk Drives during reads or writes, and a return code is passed to FLEX in the standard format with the error indicated. Usually, FLEX will report the error with a 'DRIVES NOT READY' message.
6. SETABLE DRIVE SEEK RATE - A Utility Program and User Information is included to allow setting the Disk-Drive stepping rate, either through the STARTUP.TXT or by appending FLEX.SYS.
7. DOUBLE DENSITY SPEED UP - A small 'bug' was found in the Double Density logic that was causing 'silent' read errors to slow down Double Density operations. This has been corrected.
8. ENHANCED NEWDISK - The NEWDISK Program has been slightly modified for more accurate formatting of Double Density disks. Also, it now displays information showing what processing is being performed.
9. NEW VIDEO ROUTINES - The high-resolution Video Routines have been completely rewritten to increase the display speed and add new features. These include CT-82 compatible Control Codes and display functions which provide Full X-Y Cursor Control, Erase to End of Line, Erase to End of Screen, Home Cursor, a Bell, Reverse Video character and screen, either Block or Underline Cursor, and a single Source Program which can be easily modified for video formats other than the standard 51x24 Display.
10. RSREAD - A new Utility which allows the transfer of Radio Shack Disk Basic program and data files to FLEX format disks.
11. RSWRITE - Allows writing to a Radio Shack formatted Disk from FLEX.
12. CCBASIC - An entirely new method of executing Radio Shack BASIC and EXTENDED BASIC has been developed. Basic is executed very quickly, and patched to give some additional features. The automatically installed patches allow the Saving and Loading of Basic Programs from a FLEX Disk. Also, Binary Programs can be Loaded and Saved using the standard FLEX Format for binary files. This allows Machine-Language programs to be developed using the normal FLEX Assembler, and then loaded directly into Basic; while programs saved with the Save feature are fully FLEX compatable.
13. NEW 'BOOT' PROCEDURE - A new method of developing the 'Boot Disk' is provided, which allows ANY Format FLEX Disk to be used to 'boot' FLEX, and be used as the normal System Disk (i.e., a Single Disk Boot system). A special program is provided which prepares a portion of the FLEX Disk so that Color Disk Basic sees it's own format for the BASIC Program which is used to 'RUN "FLEX"'. The FLEX Disk can be Single or Double Sided or Density (the only requirement is that the MAKESYS.CMD must be in Drive 0, and that it be used on a NEWLY formatted FLEX Disk. Also, FLEX.SYS does not have to be LINKed; the 'Boot Routine' looks to the Directory for FLEX.SYS).

Well, there you have it. I join Steve in saying "I hope that you enjoy the power and FLEXibility of FLEX9 on your Color Computer."

We have been criticized for not advertising this earlier, but we stand by our requirement that WE DON'T ADVERTISE IT until WE HAVE IT READY AND HAVE CHECKED IT OUT -- Version 2.0 is WORTH IT. As I stated earlier, THIS Flex Conversion is SOLID, and we now have the Base System to the point that Programs interfaced with it will not go out of date through updates, etc. We have mentioned numerous programs that we have run with this Conversion in the past, and we can now add TSC's new FORTRAN to the list: the only problem so far is that, like PASCAL and some of the others, it does not like a System=ALL Drive Assignment; otherwise, it runs fine (and thank goodness for 64K - that Compiler is BIG).

POWERFUL BUSINESS PROGRAMS
on the Color Computer FLEX SYSTEMS

We have been talking about the Software Development tools available for use on the Color Computer when operating under the FLEX Operating System. This month we'll look at some of the Business Programs that run with the FLEX Conversions.

A couple of months ago I ran through the operation of Pete Starks' SPELL 'N FIX Dictionary Program written for the Color Computer Disk Basic System. STAR KITS also has the same program running under FLEX, and it works great with the Color Computer FLEX Conversions. Another excellent Dictionary Program which operates with no problem with these FLEX Conversions is the STYLOGRAPH SPELLING CHECKER from STYLO SYSTEMS. The STYLOGRAPH Word Processing System is presently being configured for the DATA-COMP FLEX Conversion and it's 51x24 Display Screen, but the Spelling Checker does not require the massive Display manipulations of the Word Processor, so it only uses the FLEX I/O Vectors. This means that it works with no problems on these Conversions. It is an extremely fast and easy to use Dictionary Program, which works with any normal FLEX Text File.

Another extremely powerful Business Type Program is DYNACALC, sold by Computer Systems Center (CSC). This is the 6809 FLEX Version of the popular Visicalc program, and all commands and operations are the same as those used by Visicalc. This is probably the most popular piece of Software ever written for Computers, and has been written to run on just about every Computer System built. DYNACALC fills a large void by providing it on the FLEX Operating System, and the 6809 makes it even more powerful. Also, unlike many of the 'Brand X' Computer Systems, the FLEX Text File system is so standard that almost all FLEX Software maintains this format for output; including DYNACALC. It is a simple matter to take a DYNACALC Text File and mold it into a super report with the FLEX Based Text or Word Processors. DYNACALC has all of the features available with any of the other versions of Visicalc, including Sorting, Lookup, Index, Average, Choose, Count, Trig. and Log Functions, etc.

The obvious use for DYNACALC is in the area of Business Budgeting and Financial Forecasting, which is where the Spread Sheet Systems were originally developed. But, with the Computerization of this concept, and the flexibility and power they provide, the concept has been developed into one of the most powerful Tools available to the Businessman. DYNACALC provides the basic functions which, when properly combined, allow almost any type of computation, including Statistical, Arithmetic, Trigonometric, Tabular, etc. The capabilities of this type of program is the reason for its' popularity. The use of DYNACALC is only limited by the Operators imagination. It can be used for Inventory Control through Income Tax Preparation. Computer Systems Center has a version of DYNACALC which has been written for the DATA-COMP FLEX Conversion and it's 51x24 Display Screen. Since DYNACALC is an extremely Display Intensive program, they even provide the V51x24.CMD and Source Code with the Program. This insures that it will work with your System, no matter what we do with updating versions.

The other Major Business Program that will run on your Color Computer FLEX System I want to discuss is the Data Management Systems. There are several available for the FLEX Operating System, as you can see in leafing through the '68' Micro Journal. Each has it's advantages and disadvantages; some of the more powerful Data Base Management Systems advertised just can not be run on 5 1/4" Disk Drives. I have been working with the WESTCHESTER Applied Business Systems DMS2/VM Data Management System for the past couple of months on the Color Computer. DMS was the 'chosen' system for two reasons; Version 2.1 makes use of FLEX I/O Vectors only, insuring that it would run on the Color Computer FLEX Conversions, and it is a Machine Language Program, so it fits easily on single sided 5 1/4" Disks. We are presently working with UDRI (Universal Data Research inc.) on getting their System operational on this FLEX Conversion also.

Data Base Management systems are another extremely powerful type of Business Software. They usually consist of several different programs which are designed to work with the main Data Management program. For example, Westchester's Data Base Management System consists of a 'nucleus' program called DMS.CMD. This is initially called up, and then all of the other Programs work WITH it. The concept is a little like FLEX itself; the Operating System is 'on line' and all other programs work though it. Once DMS is 'on line', the other programs are called as required. These are DEFINE (which is used to 'define' the Data Base), UPDATE (which is used to input, change, add to, etc., data into the previously defined Data Base), GENER (for 'generate' - the real heart of the DMS system. This takes English Language Instructions from the Operator and manipulates the Data Files to produce the output desired.), and FORMAT (which allows the user to custom design the output format for specific reports, etc.). DMS consists of over 200 arithmetic, data manipulation, disk, and terminal and printer I/O functions.

A good Data Base Management system provides the power and flexibility to develop almost any type of Business Software. An obvious use of a Data Base Management system would be an Inventory System where you set each record up to include a Part No., Price, Discount, Quantity on Hand, Min. Quantity before Reordering, Part Location, etc. This would all be specified with the DEFINE Program. You would then use the UPDATE Program to enter the information into the records; one record per Part No. Once all of this information has been entered, the GENER Program would be used to 'manipulate' this Data to provide the information required. The primary Data Base would probably be arranged by Part No. so that a formatted UPDATE program could be used as a 'Cash Register' type of input program used in conjunction with the FORMAT Program to provide an automatic 'Sales Ticket' for a customer purchase that would automatically take care of pricing and inventory adjustment when the Part No. and Quantity Sold was entered. At the end of the day, the GENER Program would be used to locate any parts that needed to be restocked. What happens at inventory time? Again, call up GENER and sort the Data Base by Location. Now, if your Parts Bins were set up properly, all you have to do is walk down the rows and count parts. Not only that, but UPDATing the inventory is simple because the Data in this File is in exactly the same order that it is on the Inventory Sheets. When any changes are made to this File, they are automatically made to the Part No. ordered File because they are really one and the same Data Base; it has just been presented to you in a different order.

Westchester's DMS has several features which make it a useful System. The GENER program contains 'calculation' capabilities, which allows the System to be the basis for a full Accounting System, for instance. You can go into a Mailing List and find any Names within a specified Zip Code range. You have the capabilities of manipulating Data based on 'greater than', 'equal to', 'less than', or any combination of them. DMS's record structure is such that you can work with them using the normal FLEX Utilities; you can COPY a Data File to another Disk without losing it's structure, or DELETE a specific File without destroying the whole structure. I have only skimmed the surface of the possibilities in using a Data Base Management System, and hope that I have provided more 'ideas' that 'confusion'. This has been a general discussion, and is only intended to show a little of the potential of a good Data Management System. Suffice it to say that I haven't even skimmed the surface of the POSSIBILITIES, let alone provide specific answers.

What does all of this mean? Basically, as any good Computer System salesman will tell you, you can solve ANY Business Problem with a combination of the three Software Packages mentioned above; Word Processor, DYNACALC, and a good Data Base Management System. ALL of these will be available for the Color Computer operating on the FLEX Operating System shortly. But, it is not a matter of just running the program. Each of these Programs are extremely powerful, and require some thought and planning to make good use of them. They are really just like a High Level Programming Language; they are 'BUSINESS' Programming Languages. A good Word Processor normally must be used many hours before you begin to get the feel of it, and are able to use it effectively. Programs like DYNACALC and DMS require as much thought and planning to get them set up correctly as does writing a Program in BASIC or PASCAL. They are the basis of a good Business Program, but will only accomplish what you set them up to do. We have Columns on different Programming Languages; I would like to hear from those of you that are 'programming' in the Business Languages.

I apologize if the preceding has not been very 'coherent'; the press of beginning a new job and getting my 'System' back on line has delayed this to the point that Don is going to have to 'hold the presses' to get this in the Magazine now. As I stated before, this was presented more as a 'thought provoker' than as a 'problem solver', so get the gray matter churning and let me hear from you. (This does not only apply to the Color Computer Users, either. Let's hear it from you 'Big System' Users, too.) With that, I'll get this thing run through the Dictionary and headed North. Till next month, --- RLN ---

Publishers Note: As indicated above Bob has gone on to "greener" pastures; and I and all your friends here at 68 Micro Journal and elsewhere wish you Bob, and your family, all the best that life can bring!!! We all certainly miss your presence and I especially miss the 'great' way you brewed up the coffee, in addition to the many tasks that you so cheerfully undertook and did such a fine job of.

All letters and such for Bobs attention should be addressed to the 68 Micro Journal address, as shown on page 2. I can assure he will receive them promptly.

Again I want to thank Bob for all that he has done and for his kind assurance that he "WILL" continue to share his efforts with us all here in the pages of 68 Micro Journal. THANKS BOB!!!!!!

DMW - - -

"C" User Notes

Norm Cosmo
3 Pryor Road
Natick, MA 01760

Seems a rather nasty accident occured to the reviews published in C NOTES 5. As submitted, there was a review of the Intersoft Version 1 compiler. Unfortunately, it happened to come after a synopsis of another compiler that really wasn't in good enough shape to review yet. Don and I discussed this other compiler and felt that it was better not to say any thing about it just yet. Unfortunately when it was cut from the article so did the Intersoft review. My apologies to Intersoft for the error.

So here is the review of the Intersoft compiler as it would, and should, have appeared in C NOTES 5 of the August issue.

INTERSOFT -- version 1.0

The Intersoft compiler was the second Small C compiler to hit the 6809 market. It covers essentially the same subset of the language as the Word's Worth compiler does. It also has a rather complete, but very big, runtime package. The package includes

C.CMD	The compiler.
STDIO.H	The standard header file.
RUNLIB.A	The 16 bit arithmetic package.
CSUPP.C	The I/O and support package.
ASMSUPP.A	The low level assembler support.

The standard functions are supplied in C source code with one file for each function. The runtime support package is big because it supports such things like command line parsing into C strings and I/O redirection to files.

As with all the other Small C derivatives, this compiler is invoked from FLEX without arguments

```
++C
```

The compiler then prompts the user with the following

```
Do you wish c-test to appear?
Do you want globals to be defined?
Starting number for labels?
input file name?
Output file name?
```

If no output file is specified, the output will go to the terminal. After the first file is processed, the compiler asks for another input file. This keeps up until you enter just <cr>, at which point the compiler finishes up and returns to FLEX. The next step is to assemble the output file from the compiler and you're done.

Globals must always declared in any file where they will be referenced. The answer to the question determines whether or not they actually get defined (allocated) in the assembler code with the appropriately sized RMB, which must occur only once in the assembly process.

The starting number is the number that will be used for labels generated by the compiler. These labels are of the form "CCnnnn". You get to specify the value of nnnn, which defaults to 0 if a <cr> is entered.

Compiling separate modules, and adding in the functions from the standard library becomes tricky with the Intersoft (and later, as we will see, with the Dugger compiler). It is also at this point where I get into my biggest beef with Intersoft!

The manual details a rather nice technique for assembling a bunch of different assembler source files. I have mentioned it in a previous C column. I am sure many of you who are heavily into assembler coding have used the technique already. Essentially, you create what could be called a BUILD file. This is nothing more than a file for the assembler that contains an LIB for each of the required modules and an END directive that directs the transfer address to the C program initialization code. The assembler then pulls in all the modules that it needs.

So why the flak? As I said earlier, the compiler prompts you for the starting label. What you get to pick is the value of nnnn. Now consider the fact that all the standard library functions are supplied as C sources. One option is to compile them separately, with a unique starting label. the question is, can I start the library at, say, 8000 and increment by 100 or so? You don't know ahead of time. Some functions can be quite big.

Suddenly you remember that the compiler keeps asking for input files. That's it, you'll simply make one massive assembler file of all the functions, right? Wrong. Each of the function C sources does a "#include stdio.h" and what you get is a bunch of errors for multiply defined symbols.

Finally, since you're reasonably astute (I wasn't for a while), you realize that you must merge groups of similar functions together into one C file and edit out the multiple #INCLUDES. You then compile these files into assembler code using very widely spaced starting C addresses and you now have a series of assembler library sources. Whew, and this is what the neophyte must do just to get started.

The function library, in assembler, ends up taking quite a bit of disk space. The idea of a separate library disk that I mentioned in the Word's Worth review is VERY applicable here, although is does require more work initially. Again, it involves making each function into a separate file. My only hint on the labeling would be to compile a function, then look at its ending label. Compile the next function with a starting label somewhat higher.

Your programs in Intersoft will probably be quite big compared to the other compilers. If you want to keep their size to a minimum, then you definitely must bring in ONLY those functions that are needed to make the program complete. Do it with the BUILD file since you would only make the build file once for each C program.

I must admit that my other big gripe is a purist's nitpicking. As a 6800 user I took my share of jibes from the Z80 crowd. Well now I've got the best eight bit'er on the block, and the language to go with it. What do I find when I look at the code produced by this version of the compiler? An 8080 in 6809 clothing. The original Small C compiler generated code for a hypothetical two register machine based on the 8080. Intersoft carried that into their 6809 compiler! The result is that the code produced by compiler for a program is rather copious to say the least.

The net result was that I ran some test programs through the compiler to check out the code against the C standard. A thankless task. I have actually only run three programs using this compiler; their copy program supplied with the package, one of my own utilities and the Eratosthenes Selve Prime Number program. I encountered no bugs other than a problem with their

copy program over how to input the EOF (^Z) from the keyboard. I believe this has been rectified.

There were some minor aggrevations and oversights with some of the standard function. For example, Intersoft made a C compiler for the TRS80 before the 6809 version. The TRS80 does not have a complete keyboard. So printf() varies from the accepted norm in requiring that you use "*" instead of the backslash for escaped characters. One of the more major oversights was that fopen() dynamically allocates the required FCB space from free memory when it tries to open a file. If the file could not be opened, the FCB space is not freed up. The same is true for the fclose(). If you have a program that allows you to repeated open and close files, then it is concelvable (but not likely) that you slowly chew up a lot of free memory. At any rate, it isn't a very nice situation.

Well, Intersoft has pretty much taken it on the chin so far, but there are some bright spots. As I said before, the full runtime package is rather big, but it has all the features mentioned for Introl's $RUNFULL. And the function library is quite complete.

The manual is very well done, in format, style and clarity. They go into great detail on the hypothetical machine that the compiler produces code for. Many useful hints are given about using the compiler, although not about generating a library easily. They also give you listings of all the functions as well as describe what they do. This is very useful to someone who might not have a hard copy device readily available.

Now for some good news. You can't buy version 1 anymore. It's been replaced by version 2.5. Besides adding some more features, they have given up trying to make a 6809 behave like an 8080. I had a talk with Intersoft recently. They claim the code of version 2.5 is much much more efficient. And best of all, the price has stayed at a very reasonable $100. I don't actually have a copy of version 2.5. Like I said this was done over the phone, but I have no reason to doubt them. I may disagree with some of their implementation choices, but other than the work they put you through to get the library, and the bugs in the file functions, they seem to do a thorough job.

So the rating. I have a hard time bringing myself to use version 1, but that's a personal preference. The FCB bugs are there, but not really likely to ever bother you. On the other hand getting your first program up is a REAL chore. So I give it somewhere between an A and AA. If what they say about version 2.5 is real, and if they've smoothed over the library functions and how you get a real assembler library, then they'll do better on the next review.

Here is the output of the Intersoft version 1 compiler. The register usage is

```
D    -- The primary register
Sec  -- The secondary register, kept in memory
S    -- The system and program variable stack
X,Y,U -- General registers
```

The following example explains CCINDI.

Intersoft code

```
leay   6,s
tfr    y,d
lbsr   CCINDI  -->  CCINDI tfr    d,x
                    ldd    ,x
                    rts
```

6809 code

```
leay 6,s
ldd  ,y
```

```
*     Intersoft C Compiler Version 2.0
*#define TRUE 1
*#define FALSE 0
*#define NULL 0
*
*streq(s1, s2)
STREQ equ *

*   char s1[], s2[];
* {
*   int i;
 leas -2,s
*
*   i = 0;
 leay 0,s
 tfr y,d
 pshs d
 ldd #0
 std [0,s++]
*   while (s1[i] == s2[i])
CC2 equ *

 leay 6,s
 tfr y,d
 lbsr CCINDI
 pshs d
 leay 2,s
 tfr y,d
 lbsr CCINDI
 puls x
 stx sec
 addd sec
 lbsr CCINDC
 pshs d
 leay 6,s
 tfr y,d
 lbsr CCINDI
 pshs d
 leay 4,s
 tfr y,d
 lbsr CCINDI
 puls x
 stx sec
 addd sec
 lbsr CCINDC
 puls x
 stx sec
 lbsr CCINDI
 pshs d
 leay 2,s
 tfr y,d
 pshs d
 lbsr CCINDI
 addd #1
 std [0,s++]
 subd #1
 puls x
 stx sec
 addd sec
 lbsr CCINDC
 pshs d
 ldd #0
 puls x
 stx sec
 lbsr CCEQ
 cmpd #0
 lbeq CC3
*       if (s1[i++] == NULL)
 leay 6,s
 tfr y,d
 lbsr CCINDI
 pshs d
 leay 2,s
 tfr y,d
 pshs d
 lbsr CCINDI
 addd #1
 std [0,s++]
 subd #1
 puls x
 stx sec
 addd sec
 lbsr CCINDC
 pshs d
 ldd #0
 puls x
 stx sec
 lbsr CCEQ
 cmpd #0
 lbeq CC4
*           return(TRUE);
 ldd #1
 leas 2,s
 rts
```

Continued Next Page

68XX Pioneer Dies

HAROLD MAUCH - Mr. PERCOM DIES

It was with a deep sense of loss that word came this evening of the passing of Harold Mauch, founder of PERCOM DATA CORP, originator of the "Kansas City" tape standard and early pioneer of micro-computers.

It seems that Harold became ill some weeks back and was in the hospital for 'virus' test when complications and possible "leukemia" became too much. He passed away the evening of the 30th of August, 1982.

Our heartfelt sympathy to his wife Lucy, his family and of course those who worked and associated with him daily.

With the passing of friends who we have known it seems that a little bit of us all dies each time. Harold Mauch will be missed by his family and friends as well as the entire 68XX community. He will be remembered for his many contributions to the 68XX community and the micro computer industry.

Don Williams Sr., Publish

```
*   return(FALSE);
CC4 equ *

 lbra CC2
CC3 equ *

 ldd #0
 leas 2,s
 rts
*    }
* /* 0 errors in compilation */
 end
```

STRUCTURES

C, like Pascal and some other languages, let's you define complex data structures. It does this through a device known as the structure. Let's lead into it with an example.

Suppose you were writing a (very) simple payroll program. You will need some amount of baseline data for each employee in your organization. The minimal information, which is all we shall concern ourselves with, might be

```
last name, first name, middle initial
street address, city, state, zip
date of birth, social security number
date of hire, salary;
```

In a manual system for a small business, all this information might be kept on index cards. It would probably be arranged alphabetically by last name, with one card per employee. Wouldn't it be nice if there were some mechanism in C which would, like an index card, let us carry around all the information about an employee as a single quantity. Well there is, and it is called the "struct". The definition for a struct that could be use for a such a payroll record would be

```
struct emptype
 {
   char lstnam[20], fstnam[20], midint;
   char street[30], city[30], state[2], zip[5];
   char socno[12];
   int dob[3], doh[3];
   float pay;
 };
```

The struct definition is opened with the keyword "struct". Next comes the "tag". You might have many structs in a program. You tell the compiler which one you're really talking about by refering to it with its tag. Finally, all the definitions for the structure "members" are given.

Now we can refer to the information for an employee as one unit. To use the structure in a program we will need a variable of the type "struct emptype". The declaration might look like

```
struct emptype employee;
```

The compiler, upon finding this definition, would look up the size of emptype and allocate the necessary storage.

A simple routine to enter a new employee's name into a structure might look like this

```
printf("Enter the employee's first name: );
gets(employee.fstnam);
printf("Enter the middle initial: ");
employee.midint = getc();
printf("Enter the last name: ");
gets(employee.lstnam);
```

This example points out a few elementary facts about structs. The general method of accessing a structure member is to use the struct's name joined to the member's name with a period. In other words

```
structname.membername
```

Now think back to our discussions about arrays. We can't pass an array as a whole a function. We pass a particular member by value or we can pass a pointer to one of its members, usually the first. The same holds true for a struct. What we actually passed to gets() was the addresses of employee.fstnam and employee.lstnam, since they are strings. However, because midint was a single character, using employee.midint refers to the actual value and not the address of the value.

Assuming we had fifteen employees, we might want an array of records. This would be declared

```
struct emptype employee[15];
```

We would refer to individual members as we did before with the exception that the structname would also have an index associated with it as in

```
employee[1].lstnam.
```

If we wanted to copy all the data from second record into the third we might use the following code

```
strcpy(employee.[2].fstnam, employee[1].fstnam);
strcpy(employee.[2].lstnam, employee[1].lstnam);
employee.[2].midint = employee[1].midint;
strcpy(employee.[2].street, employee[1].street);
strcpy(employee.[2].city, employee[1].city);
strcpy(employee.[2].zip, employee[1].zip);
strcpy(employee.[2].socno, employee[1].socno);
employee[2].dob[0] = employee[1].dob[0];
employee[2].dob[1] = employee[1].dob[1];
employee[2].dob[2] = employee[1].dob[2];
employee[2].doh[0] = employee[1].doh[0];
employee[2].doh[1] = employee[1].doh[1];
employee[2].doh[2] = employee[1].doh[2];
employee[2].pay = employee[1].pay;
```

Wow, that was a lot of code just to copy one struct into another. But it graphically points out another fact about structs. You can't copy from one into another just by equating their structnames. You MUST explicitly copy each member.

There is a lot more to structs that what I have shown here. But this should hold you till next month when we will go into more detail. I will also show you a way to initialize or copy structs in a much simpler fashion.

68 Micro Journal Bulletin System

Well we now have our 68 Micro Journal Bulletin System up and running, sorta. The hardware and software are all together and the system works, so now we go to the 'shake down' part of the operation.

First, I would like to THANK all those who have contributed to this project. The interfacing and modem software, all of it was written by Mark Sproul and his father. Mark also is the designer of the Thomas Instrumentation Modem board. Needless to say that this part of the system is really SUPER! The software has not thrown us one curve and the Modem board is an 'unsung' piece of hardware that has not received the accolades that it rightly deserves! We have had this board for nearly 18 months (unused) and just within the past month, while getting the system up, have we had an opportunity to use the modem board, it is simple to interface (with furnished documentation and software)

and when we sent ours to Mark to set the system up, it came back without one, RIGHT, not one required modification, update or patch. I certainly wish others designed their hardware products with as much skill and care. Tom, you are sure doing something right.

HARDWARE & SOFTWARE

The system is comprised of the following which was donated by the respective manufacturers:

febe - Mainframe - 6800 with keyboard and power supply
GIMIX - 6800 CPU board - 16K and 32K RAM boards
Thomas Instrumentation - Modem Board - Video Board
SE Media/Data Comp - 5" dual disk system
TSC - FLEX™ and assorted utilities

The system software is the Technical Systems Consultants FLEX™ for the 6800, with assorted utilities. Also on the system is a utility I wrote called INDISK. INDISK allows up to 31K of text to be downloaded from tape or disk and saves it to disk as a text file.

Operation

The system is accessed by dialing:

A/C (615) 842-6809

Did you catch that? After contact is made the system prints a short 'hello' type message and then ask for your name. We will eventually require a special code that only subscribers to 68 Micro Journal will have access to, but for the present we are not restricting access.

The commands available are:

CAT - DIR - BUILD - WORDS - FIND - INDISK - BYE

CAT - prints the files 3 across from the system directory.

DIR - is a slightly modified standard utility.

BUILD - is the standard utility and should be used for all hand entered text files (only text type files are accepted).

WORDS - is the standard WORDS utility and allows a count of the words and lines in a file, this allows you to determine if you desire to download the file.

FIND - is the standard utility and allows searches of all text type files on the system. Useful for searching the 1979, 1980 and 1981 indexes of 68 Micro Journal, which are on the system.

INDISK - is a special utility that will accept a text file of 31K bytes of data. Longer files must be split and send as additional files. Some modems will not actually receive 300 baud (which is our ONLY rate) and INDISK always echoes back the last character received, so if you are seeing dropouts on the echo back then you should either put a timing loop into your sending program or better have your system wait until a character is echoed back before sending the next character.

BYE - this just tells our system that you are finished and is a tidy and neat way of signing off, please use it.

There are a few source files and BASIC games that can be downloaded from the 68 MJCBS, be our guest, also we and others would appreciate anything you might want to leave on the system.

PLease no 'COARSE' or abusive language. We would like for the younger readers to be able to use the system without fear of unacceptable language being present. We have a 'dirty word' filter program and hope that it is not required, for if it is then it will put a cramp in the operation of the system and even may induce me to discontinue it.

Also on the system is a modem source program that is also being published in this or a coming soon issue. It is free for you to use. If you have a better one or improve on this one let me know so I can pass it along to other readers.

I am personally using the AAA Chicago (see ad this and other issues) Super Modem Program. This program has a lot of nice features. One is the command "A" which requires the program to wait for the character echo. I have modified this program to have it come up in the "T" transmit mode and in the "A" mode. Also I changed the internal program "start over" from the "ESC" $1B to a Control Z $1E, this then allows the program to pass the ESC to our system (used by FLEX™). Other than this I have used it since we started testing the 68MJCBS and find it does about anything we need to communicate with our or anyone elses modem system. Jerry at AAA Chicago has these changes and as they furnish source for this program (useable with 6800 or 6809) you should have no problem if you need a modem program.

PLease let me know if you have any suggestion or other remarks concerning our system. It is a new thing for all of us here and we can sure stand all the help we can get.

For the immediate future the system will be active mostly in the evenings, however, we will attempt to keep it online as much as we can, even during the day and if things go ok we hope that it can be left online 24 hours a day. Also we will not accept any 'business' type files from the system. Later we hope to include normal type business transactions, but due to the 'public' nature of files on the system under our present operating system it will have to be just for the fun of it.

Again, let me hear from you on anything you believe would help or improve the system. Like I said before we need all the help on this project we can get.

WINCHESTER BACK UP

We here at 68 Micro Journal have probably been using a 'Winchester' hard disk system about as long as any Standard S50 Bus user. Without the extended storage (20 megabytes plus) we could never maintain our subscribers data files as well as other large files. Even if we could do it on floppies the time wasted gets unbearable. The SWTPC Winchester disk system, actually two of them, that we use have saved us literally thousands of extra hours of file editing. They have paid for themselves many times over.

Recently we added a 128K GIMIX CMOS battery backed up system with their new 5 inch 19 megabyte disk system, it too is paying its way rapidly. So now we have on line at least two and most times three winchester systems. All of them busy all or part of the day saving us TIME which translates quite readily to cash saved. That is what it is all about, keeping the books in the black; without our Standard S50 Bus computers we would have to employ at least 10 additional souls. Sure does not take a math genius to see that the big disk systems fast pay their way.

But, there has always been a pretty nasty fly buzzing about. After you get a big disk practically full or even half full (30 or 40 thousand sectors X 252 characters per sector) you have a lot of stored data. Now if everything was perfect and mechanical or electronic parts never failed or even 'coughed' there would be no fly to worry about. But, things being what they are we worry.

Having had to reenter thousands of lines of data to a mailing list is not the most enjoyable way to spend a

couple of weeks. We (I) have! Not that our failures have been excessive it is just that we have had failures, as any device can. The occasional glitch I expect. However, the better prepared I am to ward off any big redo because of the failure, ours or the machines, the better off I am.

Up to now we have had no simple or easy way to 'back off' our winchester system to anything. We have from time to time 'newdisked' a bunch of 8 inch disks and then proceeded to 'back off' the winchester to 8 inch disk by the FLEX™ COPY.CMD routine. Now that takes some doing, if you have a lot of files (1,000+) on the hard disk.

First, there is the problem with files that are larger than an 8 inch disk can hold. Secondly, even if they are all shorter than say 3952 or so sectors, you still must go through the time consuming procedure of 'copying' from the hard disk to the back up disk (normally 8 inch) in the following manner:

+++ COPY 3 1 <with or without parms>

So, off you go as the computer copies off the first file, then the second and so on...until you get (after quite a few minutes) disk error "ALL AVAILABLE DISK SPACE HAS BEEN USED"! Now comes the FUN part. You first delete the last file copied to the receiving disk as you are not certain that it all got there before you got the error message, you take no chances and delete the last file. Now you have the system print out a catalog or directory of the receiving disk, in order that you know what files were copied over and especially what files you must DELETE.CMD from the hard disk. Otherwise the 'stupid' machine will just start at the top of the hard disk again. Not its fault and really not fair to call it stupid, just that this is the only way it knows, and by this time everything and most everybody is 'stupid' or worse.

Having gotten this far without 30 or 40 telephone calls, a couple from 'mother nature' and maybe one from some jerk (even with a sweet sounding female voice) telling you about their latest bargain on 7 by 5 foot real estate plots, you start the grinding DELETE process. Each one, name by name, with extensions yet. Probably about 150 or so per session, of which there will likely be 15 to 20 of before you are through. Believe me if you have never experienced this ritual you should be the first in Church next Sunday!

So, as each back up disk is made and each printout and then deleting session is finished, you fuss (or worse) fume, mutter, can't wander far because you, well, just never know, drink more coffee (or something) which makes mother nature call more often, and wonder what went wrong, you should have been a 'sanitation engineer' or something else, anything else. And finally, after a lot of hours, you are finished, or at least pray you are. So you sit back down at the system and check off each one of the programs on the source disk with the listing of programs that went to the disks on the receiving system, one by one, 1000+!!

If all went well there still may be time to catch four or five hours of sleep (you have not had any for the past couple days) before you greet a shining new day.

The above is of course extreme, and actually it is sometimes simple, sometimes worse. The point being that what is needed is a nice simple way of letting the computer do its thing without so much human interaction.

Coming soon will be a new streaming tape backup system for those using the SWTPC hard disk systems. This solves the entire problem, for SWTPC users. But, what about all of those out there who have other hard disk systems? Or perhaps those who do not need the speed (about 10 minutes) to back off 20 megabytes and have an 8 inch disk drive on line. Why not software?

We here at 68 Micro Journal have long looked about for a software package to do just that. Finally we have found what we believe to be the answer. Not as fast as tape back up but certainly less expensive. It consist of three programs. They work as follows:

1. LBACKUP.CMD - this command is used first to copy off the hard disk all files, by name, that will not fit on a single disk. As each data file is copied off the receiving disk is checked for the amount of free sectors remaining. If not enough is left then the bell rings (provided your terminal has a bell) and you are prompted to remove the disk being written to and insert another blank (but NEW DISKED) receiving disk. This will continue until the entire file has been backed off. Don't worry we have another program to put it back on in proper order.

2. RESTORE.CMD - this program puts back to the hard disk the programs copied off by LBACKUP.CMD in the proper order. Even handles random files!

3. ABACKUP.CMD - this program is used after all large files (too big for one disk) have been backed off and then deleted from the hard disk (should not be very many). When the hard disk contains only files that are shorter than 'one disk full' it will start at the beginning of the hard disk and copy, in order, all remaining files from the hard disk to the receiving disk. When the program determines that the receiving disk cannot hold the entire file as there is not enough space remaining on the receiving disk, it rings the bell and prompts for another blank disk to be inserted. This continues until the entire hard disk has been backed off, with a minimum of fuss!! Actually all you do is feed it blank disk. Also it runs much faster than the standard COPY utility.

These programs will be made available from the Data-Comp division of CPI. The initial offering will be for FLEX™ systems only. UniFLEX™ systems will be supported in the near future, if a demand is indicated. The price should be on the order of $99.50 for the FLEX™ package, and if you have ever gone down the hard disk 'back off' road before, the price is certainly worth it. Watch for Data-Comp advertising soon, or if you need it like NOW give a call and hopefully it will be ready.

Sorry, I don't believe it!

All in all I still believe that we, Standard S50 Bus users are the best equipped group of micro-computer users around. We use the finest series of 8 bit CPU devices available and have the most dependable backplane going. The manufacturers of Standard S50 Bus equipment have maintained a level of quality that is unsurpassed in the micro-computer field. On the whole we have been dealt pretty good hands. If there is one place we need to focus attention, it is on support software, and that is fast being taken care of by a host of new software vendors. By software support I mean that type software that makes the micro look like a mini or more.

We have available to us practically every high level language for development efforts as well as applications software that does (and better than most) any chore you can name. Not a profound statement but a simple fact, 'we take backseat to none'. We have single and multi user systems, systems that will address more RAM than any of us can afford, systems that display, print, sing, talk and even call the cops if our 'pre-programmed' computer is attached to the proper wires. Systems we build and systems others build for us. Systems that run sorta slow and systems that zip along with the bigger minis. Systems that look like toys (but perform like tools) and systems that look as if they came straight out of 'Star Wars'. We have systems that forget everything if the lights go out 'even just for an instant' and systems that like the elephant never forget, that is as

long as their batteries hold a charge. Systems that cost as little as a couple hundred bucks and systems that cost ten times that plus. And yet with all these differences, and more, they all run the same software (makes life so enjoyable) and answer to our inputs in the same fashion and manner. Of course here as every place else you get what you pay for; the more utility you desire or need, the more the cost. Still we have from time to time come up a little short.

We have had too much software that 'hit' the market just a little early, eventually it all got straightened out but for some of us, we never did receive a check for 'debugging' somebodys new product. Also we have had some hardware that had a few 'hidden' surprises that quietly got sorted out. These glitches were at the least agitating and at times close to disastrous, yet we still came out better than the fellows with the 'other' systems. Maybe we did have too many 'versions' but at least the vendors reacted and got things the way they should have been even after we started using the product. All in all we have not had it too rough, I know personally users of some the other systems and they not only had the same type problems we did, but they had it, in practically every instance, far worse. Nothing like spending a fist full of hard earned cash for a system that does practically nothing right and then when all remedies have failed, you can't get your cash back because the outfit who made it sort of drifted away back there somewhere. We have had very little of that sort of tragedy. We have had some small vendors fold, but on the whole our losses have been minimal (except perhaps to the unlucky soul who got tangled up with the bad guys). In fact since Sphere went under some five or six years ago I cannot think of one major manufacturer who has really let us down on the Standard S50 Bus. As for the 'others' well I can name you a dozen without drawing a breath.

Now the gist of this is that I am repeatedly asked in letters and telephone calls why we (Standard S50 Bus) types don't have or do as some of the 'Johnnie come lately' biggies do. Well the truth of the matter is that they are actually having problems that we never experienced, thank goodness! Also if you really examine the capabilities of their equipment and ours, you will see that our systems do much more, in a more professional manner, with far less grief. In other words for the amount of real work accomplished we experience a lot less headaches. And that my dear friends should be the bottom line.

If you really think that things are so much brighter with a 'biggie' system, then contemplate what it would be like with say a screen with 16 lines by 40 characters, all upper case, having to purchase (at a very unreasonable price) a new disk system just to go from single sided to double sided disk (with all the attending annoying side effects), most times no source with any software packages purchased, having to reconfigure your entire system each time a different program is run (even though the software all came from the same vendor), no technical information furnished with hardware purchases (including the computer) and when you do buy the service manuals, the very information you need is 'black box' priority information, so back goes the whole shooting match if anything ever goes down. I DO NOT know of one, no not one, Standard S50 Bus vendor who makes or sells hardware, that does not furnish technical information at the time of purchase. Admittedly some of it is not as complete as it could or should be but it sure beats getting nothing but a telephone number to call for a 'return authorization' for repairs (only God knows how long).

Maybe this is all just a bit of wasted space but I KNOW what it is like on the other side and I am certainly thankful that I started out with the 6800 way back when it was a rumor. Not just for the 68X CPU, even though now it is common knowledge in the industry that the 6809 is the finest 8 bit micro going, but because those manufacturers and software types who gave us the basic systems and software have hung in there and not drifted away somewhere back there. And don't forget that there is a mountain of 'old first generation' 68XX computers, SWTPC, GIMIX and SSB who with just a little 'fixing' here and there run like the latest machines (well nearly). Tell me about any of the others who can say as much, ANY! So if you try to tell me about some of those others; well sorry but I don't believe it.

DMW - - -

splm - UniFLEX

I received a package with a preliminary copy of three fine software packages, soon to be available for UniFLEX™.

First, there will available a UniFLEX version of SPL/M. This was a very popular 6800 compiler (native code) that combines feature of BASIC, PASCAL, C and of course PL1. I have also recently received the 6809 FLEX version of SPL/M and it appears to be even better than the 6800 version. Both will be advertised soon in 68 Micro Journal.

Also included for UniFLEX was a "debug" package similar to the SIGMON for the 6800 and is useful for debugging UniFLEX SPL/M programs.

Last, but certainly not least was a spelling checker for UniFLEX with a 26,000 + word dictionary.

These programs are in the final stages before availability, but it appears that they will become very popular for UniFLEX users as well as OS9 users, as programs may be compiled (SPL/M) on a UniFLEX machine and run on an OS9 computer.

Information concerning these as well as other software can be had by contacting:

Orville Stoll III
7611 Granada Dr.
Buena Park, CA 90621

DMW - - -

C O L O R C L I N I C

TONY DISTEFANO
4680-18 Street
Laval West QUE
Canada H7R 2P9

I received a letter from Bill West of Audubon PA. He seems to have a problem with the version 1.1 and 1.0, here is his letter:

Dear Tony,
In June'82 I received a 4K Color Computer as a graduation gift from Medical school and within 1 week had it upgraded to 32k with Extended Basic. Since that time it has been returned to a Radio Shack store three times for repair - the first time the SAM chip was replaced; the second time the VDG chip was replaced; the third time the cassette relay was replaced. However, it seems to be functioning OK at the present time. After the original upgrade the screen display read "EXTENDED BASIC 1.1", but since the last two repairs the screen display read "EXTENDED BASIC 1.0" on power up. The computer is recognizing the 32k RAM as evidenced by PRINT MEM and when questioned, the RS serviceman replied that there was no difference between the 1.0 EXTENDED BASIC ROM and the 1.1 EXT. BASIC ROM. From my reading

of various articles in 80 MICRO and other journals, I get the impression that there is a difference between the 1.0 and the 1.1 ROMs but I haven't been able to determine the advantage(s) or disadvantage(s) of eighter ROM. So - my question is, should I take the computer back to Radio Shack and demand that they restore it to EXTENDED BASIC 1.1 or doesn't it matter? I do plan to upgrade the system to Disk and possibly 64k with Flex along with a printer in the near future, but by that time my warranty will have expired. What should I DO?

Thanks
Bill West, D.O.

To answer Bill's question, it is necessary to explain what you see on the screen when you turn the computer on. When power is applied to the computer, the CPU jumps to a routine in Basic ROM that is called a "COLD START". The CPU then proceds to initialize the PIA, SAM and VDG chips. It then checks to see how much memory is available and sets the pointers acordingly. It sets up variables like baud rate and line delays and many more. It then checks to see if the EXTENDED BASIC ROM is present. If it is, it jumps to the EXTENDED BASIC cold start. Now the EXTENDED BASIC ROM does its own cold start routine like set up the graphic pages and so forth. IF there happens to be a DISK controler in the cartridge slot, then the DISK BASIC ROM does its cold start routine. Have patience all this is leading up to the reason for all this.

Going back to just plain BASIC, it tests for the EXTENDED BASIC ROM, if it isn't there, it goes on to clear the screen and put up the logo. This is where the versions start to come in. The screen will show "COLOR BASIC 1.0" or " COLOR BASIC 1.1 ", because there are two versions of Color Basic. When EXTENDED BASIC is present and does its cold start it will print "EXTENDED COLOR BASIC 1.0". Since there is only one version (As far as I know, if anyone has seen another version please contact me) of Extended Basic, it will always show 1.0, no matter what version of regular Basic you have. The same is true with "DISK EXTENDED COLOR BASIC". There is only one version of the DOS (Disk Operating System) too, 1.0. This means that if you have Extended Basic or Disk Extended Basic you cannot tell what version of regular Basic you have.

Well, there are a few ways of knowing. One is, if you have a Radio Shack installed 32k package or you bought a 32k computer you have the 1.1 version of Basic. Another way of finding out is by typing in this command; " EXEC 41175 " and hitting the enter key. This will put on the screen the logo for the Color basic and of course the version. The easiest way to tell is to plug in the joystick and press the "FIRE" button. If a string of characters appear on the screen then you have the 1.0 version.

Now that you know how to tell what version of Basic you have, I will discribe the main differences between the 1.1 and the old 1.0 Basic. The most important difference is that the 1.1 can detect the presence of the RS 32k chips and 64k chips. The 1.0 could not do this. The chips would not be addressed properly. Another change made, is in the RS-232. The 1.1 version sends out 8 bits while the 1.0 version only sent out 7. This made it possible to send graphic dumps without extra software. When the "FIRE" button was pressed in the old version, a string of characters would appear on the screen. The string no longer appears in the new version. This might present a problem to software written for the old version. A call to "POLCAT" will not return a value when you press the "FIRE" button. Therefore you will have to hit a key to get a response. There are other minor differences, but not worthwhile mentioning. Besides, when you buy a computer with the 1.1 version installed, there is a notice explaining the differences.

Bill, I think that the explanation above answers most of your questions. If you want to get FLEX make sure that you indeed have the version 1.1 of the BASIC ROM. You cannot do without it if you plan to put in 64k chips.

Home Acct Prog

Part 1!!!

ERNEST STEVE WATSON
11701 ST. CHARLES BLVD.
LITTLE ROCK, ARKANSAS 72211

OR

F. DALE BRADY
7729 BRADLEY DRIVE
LITTLE ROCK, ARKANSAS 72209

```
0 REM PRTSTAT.BAS
20 CL$=CHR$(27)+"E":W=60
30 PRINTCL$
40 OPENOLD"1.YEAR"AS1
50 GET#1,RECORD1
60 FIELD#1,2ASTN$,2ASCY$,3ASCM$,3ASPM$,40ASCN$
70 Y%=CVT$%(CY$)
80 P$=PM$:M$=CM$:N$=CN$
90 CLOSE1
100 PRINTCL$
110 OPENOLD"1."+P$+".GL" AS 1
120 PRINT:PRINTTAB(W/2-8);"Print Statement"
130 PRINT:PRINT"Enter END to return to MENU"
140 INPUT"Enter Account Number",F1$
150 IFF1$="END"THENCHAIN"MENU.BAS"
160 F1=VAL(F1$)
170 IFF1<100 OR F1>900 THEN140
180 PRINT:PRINT
190 PRINT"Do you want PRINTER output (Y/N)? ";
    :AN$=INCH$(0):PRINT
200 GET#1,RECORD1:FIELD#1,2AS1$:I=CVT$%(I$)
210 FORIZ=1TOI:RZ=IZ/8:SZ=IZ-(RZ*8)
220 GET#1,RECORDRZ+1
230 FIELD#1,SZ*30AS1$,2AS6N$,20AS6M$,8AS6T$
240 IFCVT$%(GN$)=F1THEN260ELSENEXTIZ
250 PRINTCHR$(7);"ERROR ACC# ";F1;" NOT FOUND
    REENTER":GOTO130
260 A=CVT$%(6N$):A$=6M$:A9=CVT$F(6T$)
270 CLOSE1
280 ONERROR GOTO620
290 OPENOLD"1."+M$ AS2
300 PRINTCL$
310 IFAN$="Y"THENPOKE40972,0:EXEC,"TTYSET,EJ=21,
    DP=45,PS=N,NL=25"
320 PRINT:PRINT
330 PRINT:PRINTTAB(W/2-20);N$
340 PRINT:PRINT
350 PRINT"Acct#";A;" Statement for ";A$
360 PRINT
370 AI=A1+A9
380 PRINT"Balance brought forward to ";M$;",";YZ;
390 PRINTUSING"$$#,###.##",A9
400 PRINT
410 GET#2
420 FORSZ=0TO4
430 FIELD#2,SZ*50AS2$,2AST0$,2ASTC$,2ASTN$,10ASTY$,
    26ASTP$,8ASTA$
440 DZ=CVT$%(TD$):CZ=CVT$%(TC$):NZ=CVT$%(TN$)
    :A=CVT$F(TA$)
```

```
450 IFDZ=FI THEN 480
460 IFCZ=F1 THEN 510
470 GOTO 530
480 A1=A1+A
490 PRINTUSING'\2345\ @@@@@
      \12345678901234567890123456\$$$,$$$.@@',TY$,
      NZ,TP$,A
500 GOTO 530
510 A1=A1-A
520 PRINTUSING'\2345\ @@@@@
      \12345678901234567890123456\$$$,$$$.@@',TY$,
      NZ,TP$,A1(-1)
530 NEXTSZ:GOTO410
540 CLOSE 2
550 PRINT:PRINTTAB(35);
560 PRINTUSING'TOTAL $$@,@@@.@@',A1
570 IFAN$="Y"THENPOKE40972,1:EXEC,"TTYSET.EJ=0,
      DP=24,PS=Y,NL=0"
580 PRINT"Hit any key to continue? ";:AN$=INCH$(0)
590 PRINT:PRINT"Do you want to run another ACCOUNT
      LEDGER (Y/N)? ";:AN$=INCH$(0)
600 IFAN$="Y"THENAI=0:GOTDI00
610 IFAN$="N"THEN CHAIN"MENU.BAS" ELSE GOTO590
620 IFERR=8THENRESUME540
630 IFERR=4THENCLOSE2:PRINT"NO TRANSACTIONS FOR ";
      M$:RESUME570
640 ONERRORGOTO
```

```
0 REM TRAYEAR.BAS
20 REM THIS PROGRAM IS CALLED FROM POSTGL.BAS
25 PRINTCHR$(27);"E"
30 OPENOLD"1.YEAR"AS1
40 GET#1,RECORD1
50 FIELD#1,2ASTN$,2ASCY$,3ASCM$,3ASPM$
60 XZ=CVT$Z(TN$):REM # IN YEAR FILE
70 M$=CM$:REM CURRENT MONTH
80 P$=PM$:REM LAST POSTING MO
90 CLOSE 1
92 PRINT"TAANSFERING ";M$;" DATA TO YEAR FILE"
93 PRINT:PRINT
95 PRINT"NUMBER OF TRANSACTIONS IN YEAR TO DATE
     FILE =";XZ
100 REM GET SIZE OF TRANS FILE
110 OPENOLD"1."+M$AS1
120 GET#1,RECORD1
130 FIELD#1,2AS2$:TZ=CVT$Z(2$)
150 DIMDZ(TZ),CZ(TZ),CNZ(TZ),D$(TZ),P$(TZ),A(TZ)
170 PRINT"NUMBER OF TRANSACTIONS IN ";M$;" FILE =";TZ
180 REM READ IN TRANS FILE
190 GET#1,RECORD1:GOTO210
200 GET#1
210 FORSZ=0TO4
220 FIELD#1,SZ#50ASZ$,2ASTD$,2ASTC$,2ASTN$,10ASTY$,
      26ASTP$,8ASTA$
230 IFCVT$Z(TD$)<100THEN280
240 JZ=JZ+1
250 DZ(JZ)=CVT$Z(TD$):CZ(JZ)=CVT$Z(TC$)
      :CNZ(JZ)=CVT$Z(TN$)
260 D$(JZ)=TY$:P$(JZ)=TP$:A(JZ)=CVT$F(TA$)
270 IFJZ=TZTHEN300
280 NEXTSZ
290 GOTO200
300 CLOSE1
310 OPENOLD"1.YEAR"AS1
320 FORIZ=XZ+1 TO XZ+JZ
330 IFDZ(IZ-XZ)<100 THENPRINTCHR$(7):GOTO450
340 RZ=IZ/5:SZ=IZ-(RZ#5)
350 IFSZ=0THENFIELD#1,252AS2$:LSET2$="":GOTO370
360 GET#1,RECORDRZ+1
370 FIELD#1,SZ#50ASZ$,2ASTD$,2ASTC$,2ASTN$,10ASTY$,
      26ASTP$,8ASTA$
380 LSETTD$=CVT2$(DZ(IZ-XZ))
390 LSETTC$=CVT2$(CZ(IZ-XZ))
400 LSETTN$=CVT2$(CNZ(IZ-XZ))
410 LSETTY$=D$(IZ-XZ)
420 LSETTP$=P$(IZ-XZ)
430 LSETTA$=CVTF$(A(IZ-XZ))
440 PUT#1,RECORDRZ+1
450 NEXTIZ
460 REM KILL"1."+M$+".DAT"
470 REM UPDATE YEAR
475 PRINT:PRINT"NUMBER OF TRANSACTIONS IN YEAR
      FILE=";IZ
480 XZ=IZ:DIMM$(12)
490 FORIZ=0TO12:READM$(IZ):NEXTIZ
500 DATA DEC,JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,
      OCT,NOV,DEC
510 FORIZ=0TO12:IFM$=M$(IZ)THENS20ELSENEXTIZ
      :PRINT"ERROR":END
520 P$=M$:REM NEW POSTING MONTH
530 M$=M$(IZ+1):REM NEW TRANS MONTH
540 GET#1,RECORD1
550 FIELD#1,2ASTN$,2ASCY$,3ASCM$,3ASPM$
560 LSETTN$=CVT2$(XZ):REM UPDATE NUMBER IN FILE
570 LSETCM$=M$:REM NEW MONTH
580 LSETPM$=P$:REM POSTING MONTH
590 PUT#1,RECORD1
600 CLOSE 1
610 CHAIN"MENU.BAS"
```

```
0 REM CHECKWTR.BAS
20 CL$=CHR$(22)
30 WZ=60
40 OPENOLD"1.YEAR"AS1
50 GET#1,RECORD1
60 FIELD#1,2ASTN$,2ASCY$,3ASCM$,3ASPM$,40ASCN$
70 XZ=CVT$Z(TN$):REM # IN YEAR FILE
80 YZ=CVT$Z(CY$):REM YEAR
85 YZ=YZ-1900
90 M$=CM$:REM CURRENT MONTH
100 P$=PM$:REM LAST POSTING MONTH
110 N$=CN$:REM NAME
120 CLOSE 1
130 OPENOLD"1."+M$ AS1
140 PRINTCL$
150 PRINT:PRINTTAB(WZ/2-8);"CHECK WRITER PROGRAM"
160 PRINT:PRINT"Checks will be taken from the ";M$;
      " transaction file"
170 PRINT:PRINT:PRINT"Enter END to return to MENU"
180 PRINT:INPUT"Enter Beginning Ck#....",AN$
190 IFAN$="END"THEN960
200 BZ=VAL(AN$):IFBZ<1THENPRINTCHR$(7):GOTO180
210 PRINT:INPUT"Enter Ending Ck#.......",AN$
220 IFAN$="END"THEN960
230 EZ=VAL(AN$):IFEZ<1THENPRINTCHR$(7):GOTO210
240 IFEZ<BZTHENPRINT"ENDING # SHOULD BE HIGHER THAN
      BEGINNING #":GOTO180
250 ONERRORGOTO250:IFERR=4THENPRINTCHR$(7);M$;"
      TRANSACTION FILE EMPTY":GOTO960
260 IZ=0:GET#1,RECORD1:FIELD#1,2AS2$:XZ=CVT$Z(2$)
      :GOTO280
270 GET#1
280 FORSZ=0TO4
290 ONERROR GOTO
300 FIELD#1.SZ#50ASZ$,2ASTD$,2ASTC$,2ASTN$,10ASTY$,
      26ASTP$,8ASTA$
310 IFBZ=CVT$Z(TN$)THEN350
320 IFIZ=XZ THENPRINTCL$;CHR$(7);"CHECK NUMBER ";BZ;"
NOT FOUND":GOTO170
330 IZ=IZ+1
340 NEXTSZ:GOTO270
350 NZ=CVT$Z(TN$):P$=TP$:A=CVT$F(TA$)
360 PRINTCL$;CHR$(7)
370 PRINT:PRINT:PRINT"CHECK TO BE PRINTED"
380 PRINT:PRINT
390 PRINT"CHECK#     DESCRIPTION              AMOUNT"
400 PRINT
410 PRINTUSING'@@@@@  \12345678901234567890123456\
      $$,$$$.@@',NZ,P$,A
420 PRINT:PRINT"(PUT CHECK IN PRINTER)"
430 PRINT:PRINT"IS THIS CORRECT? ";:AN$=INCH$(0)
440 IFAN$()"Y"THEN140
450 BZ=BZ+1
460 EXEC,"TTYSET,PS=N,NL=25"
470 POKE 49164,0
480 PRINT:PRINT:PRINT:PRINT:PRINT
490 PRINTTAB(30);TY$;TAB(41);YZ
500 PRINT
510 PRINTTAB(7);P$;TAB(45);
520 PRINT USING '$$@@@.@@',A:GOTO 550
530 REM ROUTINES TO EQUATE AMOUNT WITH PRINTED
      PORTION
```

```
540 REM OF THE CHECK
550 PRINT
560 REM THOUSANDS FIRST
570 A1=INT(A/1000)
580 IF A1>99 THEN GOTO 930
590 IF A1>=1 THEN Z1$=" THOUSAND "
600 IF A1<1 THEN GOTO 640
610 I=A1:GOSUB 1020
620 A1$=I$
630 REM HUNDREDS NEXT
640 A2=INT(A/100)-(10*A1)
650 IF A2>=1 THEN Z2$=" HUNDRED "
660 IF A2<1 THEN GOTO 690
670 I=A2:GOSUB 1020
680 A2$=I$
690 A3=INT(A/10)-((100*A1)+(10*A2))
700 IF A3<2 THEN GOTO 730
710 I=A3*10:GOSUB 1020
720 A3$=I$
730 A4=INT(A/1)-(((1000*A1))+((100*A2)+(10*A3)))
740 IF A3<2 AND A3>0 THEN A4=A4+10
750 IF A4<1 THEN GOTO 780
760 I=A4:GOSUB 1020
770 A4$=I$
780 IF A1$="" THEN GOTO 800
790 PRINT A1$;" ";Z1$;
800 IF A2$="" THEN GOTO 820
810 PRINT A2$;" ";Z2$;
820 IF A3$="" THEN GOTO 840
830 PRINT A3$;" ";Z3$;
840 IF A4$="" THEN GOTO 860
850 PRINT A4$;"";
860 D1=A-(INT(A))
870 D1=D1*100
880 IFD1=0 THENPRINT" AND NO/100":GOTO900
890 PRINT " AND";D1;"/100"
900 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
910 POKE 49164,1
920 EXEC,"TTYSET,PS=Y,NL=0"
930 A1=0:A2=0:A3=0:A4=0:A1$="":A2$="":A3$="":A4$=""
940 IFN%<>E%THEN260
950 CLOSE 1
960 CHAIN"MENU.BAS"
970 DATA "ONE","TWO","THREE","FOUR","FIVE","SIX",
      "SEVEN"
980 DATA "EIGHT","NINE","TEN","ELEVEN","TWELVE",
      "THIRTEEN"
990 DATA "FOURTEEN","FIFTEEN","SIXTEEN","SEVENTEEN"
1000 DATA "EIGHTEEN","NINETEEN","TWENTY","THIRTY",
      "FORTY"
1010 DATA "FIFTY","SIXTY","SEVENTY","EIGHTY","NINETY"
1020 FOR J=1 TO I
1030 READ X$
1040 NEXT J
1050 RESTORE
1060 RETURN

0 REM ADDGL.BAS
20 REM 1/28/82
30 CL$=CHR$(27)+"E"
40 W=80
50 PRINTCL$
60 PRINT:PRINTTAB(W/2-15);"ADD NEW ACCT.# TO GEN.
     LEG."
70 PRINT
80 PRINTTAB(W/6);"Enter END for MENU"
90 PRINTTAB(W/6);"How many new Account's ";:INPUTAN$
100 IFAN$="END"THENCHAIN"MENU.BAS"
110 NA%=VAL(AN$):IFNA%<>0THEN90
120 OPENOLD"1.YEAR"AS1:GET#1,RECORD1
130 FIELD#1,7AS1$,3ASPM$
140 GL$=PM$
150 CLOSE1
160 ONERRORGOTO870
170 REM GET SIZE OF GENLEG FILE AND DIM VAR.
180 OPENOLD"1."+GL$+".SL" AS1
190 GET#1,RECORD1:FIELD#1,2AS1$:X%=CVT$%(1$)
200 REM DIM VARIABLES SIZE OF GL + NEW ACCOUNTS
210 DIMN%(X%+NA%),A$(X%+NA%),A(X%+NA%)
220 IFX%=0THEN340
230 REM READ IN GEN LEG
240 GET#1,RECORD1:GOTO260
250 GET#1
260 FORS%=0TO7
270 FIELD#1,S%*30AS1$,2ASGN$,20ASGM$,8ASGT$
280 IFSM$=""ORCVT$%(GN$)<100THEN320
290 I%=I%+1
300 N%(I%)=CVT$%(SN$):A$(I%)=GM$:A(I%)=CVT$F(GT$)
310 IFI%=X%THEN340
320 NEXTS%
330 GOTO250
340 CLOSE1
350 PRINTCL$
360 PRINT:PRINT"YOU HAVE ";X%;" ACCOUNTS IN
      GENLEDGER"
370 PRINT:PRINT"YOU CAN ADD";NA%;"MORE ACCOUNTS (THIS
      TIME)."
380 PRINT:PRINT
390 PRINT"Enter END to return to MENU"
400 INPUT"Enter NEW Account number",AN$
410 IFAN$="END"THEN610
420 N%=VAL(AN$):IFN%<1000RN%>900THENPRINTCHR$(7);
      "NUMBER OUT OF RANGE":GOTO370
430 FORI%=1TOX%:IFN%=N%(I%)THEN440 ELSENEXTI%:GOTO450
440 PRINTCHR$(7);"ACCT.#";N%;" ";A$(I%);"   (IN USE
      REENTER)":GOTO400
450 INPUT"Enter NEW Account NAME  ",AN$
460 IFAN$="END"THEN610
470 A$=LEFT$(4N$,20)
480 FORI%=1TOX%:IFA$=A$(I%)THEN490ELSENEXTI%:GOTO500
490 PRINTCHR$(7);"THE NAME ";A$;" IS IN USE WITH
      ACCT#";N%(I%);" (REENTER)":PRINT:GOTO450
500 REM
510 A=0
520 PRINT:PRINT
530 PRINTUSING'### \1234567890123456789\
      $$#,###.##',NZ,A$,A
540 PRINT:PRINT"IS THIS CORRECT (Y/N)? ";
      :AN$=INCH$(0)
550 IFAN$="N"THEN350
560 IFAN$<>"Y"THEN540
570 X%=X%+1:NA%=NA%-1:REM UPDATE POINTERS
580 N%(X%)=N%:A$(X%)=A$:A(X%)=A
590 IFNA%<=0THEN610
600 SOTO350
610 PRINT:PRINT:PRINT"SORTING PLEASE WAIT"
620 I%=1
630 IFI%>=X%THEN670
640 IFN%(I%)<=N%(I%+1)THENI%=I%+1:GOTO630
650 SWAPN%(I%),N%(I%+1):SWAPA$(I%),A$(I%+1)
      :SWAPA(I%),A(I%+1)
660 I%=I%-1:IFI%<=1THEN620ELSE630
670 PRINT"UPDATING GENERAL LEDGER FILE"
680 OPENNEW"1.DUMP.GL"AS1
690 R%=0
700 FORS%=0TO7
710 FIELD#1,S%*30AS1$,2ASGN$,20ASGM$,8ASGT$
720 LSETGN$=CVT$(N%(R%+S%))
730 LSETGM$=A$(R%+S%)
740 LSETGT$=CVTF$(A(R%+S%))
750 IFR%+S%=X%THENPUT#1:GOTO790
760 NEXTS%
770 PUT#1
780 R%=R%+8:GOTO700
790 GET#1,RECORD1:REM ADD FILE SIZE
800 FIELD#1,2AS1$:LSET1$=CVT%$(X%)
810 PUT#1,RECORD1
820 CLOSE1
830 KILL"1."+GL$+".GL"
840 RENAME"1.DUMP.GL","1."+SL$+".GL"
850 PRINT"LOADING MENU"
860 CHAIN"MENU.BAS"
870 IFERR<>4THENONERRDR SOTO
880 CLOSE1:OPENNEW"1."+GL$+".GL"AS1
890 FIELD#1,2AS1$:LSET1$=CVT%$(0)
900 PUT#1,RECORD1
910 CLOSE1:RESUME180
```

```
THIS IS A SAMPLE GENERAL LEDGER

 101 .CASH, 000
```

```
103 ,CASH IN BANK, 1087.97
104 ,SAVINGS-ACCT., 0
105 ,SAVINGS-CD, 13309
106 ,SAVINGS-IRA'S, 0
107 ,MUTUAL FUNDS, 0
109 ,STOCK, 4065
111 ,JEWELRY, 2500
121 ,CARS, 5000
125 ,SILVER, 7500
131 ,REIMBURSED EXPENSES, 000
141 ,CAMERAS, 500
151 ,COMPUTER, 2097.36
161 ,TAX REFUND, 1000
171 ,WITHHELD TAXES, 000
181 ,TOOLS, 200
183 ,FURNITURE, 11307.3
191 ,HOUSE, 80000
200 ,A/C PAY.-OTHER, 112
201 ,A/C PAY.-WARD'S, 0
202 ,A/C PAY.-SEARS, 0
203 ,A/C PAY.-FWNB, 0
204 ,A/C PAY.-MC,-1200.75
205 ,A/C PAY.-PENNEY, 0
206 ,A/C PAY.-VISA,-1621
210 ,CAR LOAN,-1401.72
212 ,1ST MORT. HOME,-45972
213 ,2ND MORT. HOME, 0
214 ,STUDENT LOAN,-10000
300 ,EQUITY,-68483.16
401 ,FOOD & HOUSEHOLD, 0
403 ,FOOD OUT, 0
405 ,GAS, 0
407 ,ELECTRIC, 0
409 ,WATER, 0
411 ,TELEPHONE, 0
413 ,CLOTHING, 0
415 ,GIFTS, 0
416 ,WEDDING, 0
417 ,AUTO REPAIR, 0
419 ,GASOLINE, 0
421 ,HOME REPAIR, 0
423 ,PERSONAL CARE, 0
425 ,RECREATION, 0
427 ,EDUCATION-COLLEGE, 0
429 ,EDUCATION-CONTINUING, 0
431 ,INSURANCE-HEALTH, 0
432 ,INSURANCE-CAR & HOME, 0
433 ,INSURANCE-LIFE, 0
435 ,SUBSCRIPTIONS, 0
437 ,MEDICAL, 0
439 ,STATE INCOME TAX, 0
441 ,FEDERAL INCOME TAX, 0
443 ,LOCAL TAXES, 0
445 ,FICA TAXES, 0
447 ,MISCELLANEOUS, 0
449 ,CHARITABLE, 0
451 ,INTEREST EXPENSE, 0
453 ,POLITICAL, 0
455 ,EMPLOY.& PROF., 0
500 , AGES-YOU,-0
502 ,WAGES-ME,-0
503 ,OTHER WAGES, 0
504 ,INTEREST INCOME,-0
506 ,DIVID ND INCOME,-0
END
```

B/U RECOVER

Edgar M. Pass,Ph.D.
Computer Systems Consultants, Inc.
1454 Latta Lane,
Conyers, Ga 30207
Telephone Number (404)483-1717/4570

Introduction

A modern data management system for any computer system must take into account the inevitability of computer system and component failure. Therefore, some form of backup and recovery procedures must be incorporated into the system to attempt to maintain the integrity of the data base despite hardware, software, or human malfunction. This discussion defines some of the problems and nomenclature and provides some suggested solutions to the problems.

The management of each set of data base applications must consider the requirements on the system to determine how critical recovery is from time-required, cost-required, and data-integrity criteria, how much backup depth is required, how much hardware, software, real time, and human overhead is demanded, how sophisticated a backup and recovery system is needed, how technically competent a staff is required to implement the procedures, etc. These considerations partially dictate the final design of the data base, including the backup and recovery methods.

For example, consider the cost to a company of a failure of a business computer system during the prime working hours of a business day. In a company not particularly dependent upon computers, business would continue, perhaps with some aggravation to the employees, until the system were working again. In a company dependent upon computers, particularly a large company, business might literally grind to a halt. Idling data entry clerks, making answering of customer inquiries impossible or extremely difficult, making taking of new orders very time-consuming and error-prone, etc. In fact, most large companies would go bankrupt rather quickly without access to the business information stored in their computers. Thus, in extreme cases, the maintenance of data base integrity may truly make or break a company.

Simple Backup and Recovery Systems

The conceptually simplest form of backup procedure (besides none at all) is the performance of file or volume oriented dumps at periodic or epochal intervals. Horror stories of systems with no or insufficient backup are common and, even if they do not always reflect a particular situation accurately, they generally contain enough colloquial wisdom related to the effects of major computer system problems on businesses to indicate the potential disasters present.

Copying an entire hard disk to another, copying an entire magnetic to another, copying an entire hard disk to magnetic tape, copying an entire hard disk to multiple floppy diskettes, and copying an entire diskette to another are all examples of volume-oriented backup procedures. Often, these procedures are reasonably fast and are performed without regard to data structures on the source or target volumes, but often do not allow selective, file-oriented recovery. However, dumping an entire hard disk to floppy diskettes will require a large amount of time and a large number of diskettes; also, given the relative reliabilities of hard disks and floppy diskettes, duplicate backups may be required to ensure a high level of data integrity.

Copying one or more files from hard disk or floppy diskettes to hard disk, magnetic tape,

or floppy diskette is an example of a file-oriented backup procedure. These procedures are generally slower on a per-byte-transferred basis because they follow file structures, but may be faster since fewer bytes may be transferred and they also allow file-oriented recovery. The backup intervals may be fixed, such as daily or weekly, occasional, or epochal, based upon the completion of a processing cycle, such as general-ledger end-of-month.

System failure recovery then requires the reloading of the affected volumes or files and the reprocessing of all data base updates, in the original order, to recreate the data base as it would have been had the error not occured. The maximum time lost will be the period since the last backup. Recovery is primarily manual and the amount of time to recover is based upon the number of updates since the last valid backup. It requires the maintenance of manual audit trails which will not be used unless the system fails, and thus may not be faithfully prepared. It is subject to human error in re-entry and the very real possibility that the original problem may not have really been corrected and may re-occur during the attempted recovery. Also, depending upon the application, new inquiries and updates to the file probably will not be corrected or allowed until after the recovery procedures have been completed. Despite the potential problems, the direct backup and recovery method works well in many situations, including most program development systems, and is definitely the simplest to implement.

Audit Trails

All backup and recovery schemes (other than the simplest) require the automatic generation and use of audit trails. An audit trail provides a history of events affecting one or more data bases. The level of detail maintained depends upon the situation. A simple analogy to an audit trail is a personal check register. Every check, deposit, and void is recorded. End-of-month may also be recorded to assist in periodic reconciliation. The reconciliation process itself involves checking the bank's records with the check-holder's to determine if they have recorded the same events, and thus have the same ending balance. In addition to use their reasonably temporary use in backup and recovery, audit trails may be required on a more permanent basis by auditors, accountants, and the tax authorities for their purposes, especially for data bases with financial, fixed-asset, or inventory information.
In terms of backup and recovery, the only absolutely-essential events on an audit trail are those which update the data base. Inquiry or epochal events may be useful to include for security or historical-record purposes, but are not essential to record, with one potential exception. If a data base becomes corrupted, inquiries performed afterward may be suspect, and special actions taken against the data base, such as month-end, may require repeating. These considerations must be balanced against the additional overhead that will be imposed on the data management system by the recording of additional audit trail records.

Examples of audit trail classifications include the following, all time-and-date stamped:

beginning of programs,
end of programs,
before-image of updated records or fields,
after-image of updated records or fields,
images of accessed records or fields,
images of inquiry requests and answers,
file dumps,
file loads,
user logons,
user logoff,
I/O errors,
etc.

Given that an adequate audit trail is maintained, several types of recovery schemes are made possible by its existence. The following discussion indicates the considerations involved in designing recovery plans for various types of problems.

Failure Reason Categorization

Since the type of recovery applied to a failure at least partially depends upon the type of failure, the personnel responsible for the system containing the data base must either be technically competent to diagnose the problem (perhaps with the aid of checklists, flowcharts, decision tables, and other inanimate aids) or must be able to draw on the expertise of one that is technically competent. In practice, most failures will usually be due to a small number of recognizable classes and the operational personnel will probably become responsive to these problems very soon. One danger which must be guarded against, however, is complacency in terms of the personnel attempting to solve all problems with one method without regard to written procedures.

System Software Failures

The first step is to identify programs and transactions affected by the failure. The recovery process must restore the data base to a secure point before the software failure. If a before-image of updates is recorded on the audit trail, recovery will involve applying these images against the current data base, in reverse chronological order, to back out the changes. Alternately, after-images could be applied in forward chronological order down to the point of failure against the backup copy of the data base to reapply changes before the failure. The frequency of update operations, the size of the data base, and the frequency and method of backup will usually dictate which of these methods should be applied in a particular case.

In the technical literature, these methods are generally referred to as backward and forward recovery, for obvious reasons. In those data bases capable of implementing both types of recovery, the reason for failure will normally dictate which method should be applied in a particular case. The reason for failure will also dictate whether update and other programs may be quickly restarted without additional effort. If the system is transaction-oriented, recovery may be possible up to (but not necessarily including) the last transaction, rather than the first transaction. It may be necessary to run other programs against the data base before or after transactions may be re-entered, to ensure integrity of the data base before full operations may be allowed.

Hardware Failures

The normal worst-case failure is the loss of the use of the entire disk volume(s) housing the data base. If at all possible, the audit trail should be kept on a separate disk device

or created on tape. If both the data set and audit trail are lost, the only recovery possible will prooably be to reload the entire data base and to re-enter all data base updates entered since the last backup was made. If the audit trail is secure but the data base is lost, forward or backward recovery may be used to help minimize the manual recovery effort.

If the failure involves only the loss of the audit trail, the data base should be immediately backed-up to attempt to ensure that the worst-case failure (the loss of both the data base and the audit trail) does not occur.

A compromise alternative, often used for small systems with relatively infrequent data base updates, is to mark the updated records (with a time-and-date stamp or update cycle number). The records updated since the last partial backup operation may be periodically copied to another volume or device, adding to the previous after-images there. Since this is a relatively fast operation (compared to copying the entire data base) and reduces the overhead of maintaining an explicit audit trail, it can be done more often, effectively forming an audit trail which at any given time resides partially internal to the data base and partially external to it. In case of failure, the recovery procedure is partially automatic and partially manual, but usually cost-effective in the applicable cases.

In any case of hardware failure, an attempt must be made to locate the source of the failure and correct it, if possible, before recovery procedures are instituted. Depending upon the level of critical need for access to the data base and the level of data integrity required of the contents of the data base, duplicate hardware, from the board to the system level, may be appropriate, as may be specialized hardware, such as uninterruptable power supplies.

Application Software Failures

This type of failure is probably the most common in young systems and second only to hardware failures in mature systems. Unless the structure of the data base is corrupted or the system crashes due to the error, the problem may lie dormant in the data base for a considerable period of time. If the problem is detected quickly, recovery procedures as described above may be used to return the data base to a point in time before the error, the problem may be corrected, and forward recovery may be used to re-update the data base correctly and automatically. If the problem is not detected quickly, no pre-defined, automatic, procedures may suffice, since backups may not be old enough to pre-date the error or too much effort, time, and expense may be required to use normal procedures to purge the data base of the problem. Manual procedures and special-purpose programs may be required to locate and correct the corrupted records.

In order to limit the time and scope of errors, several concepts often used by auditors may be integrated into the data base. Hierarchial control totals involve the use of records hidden in the data base which carry accumulators and counters representing some sub-tree of the data base, normally based upon a portion of the primary key. Then if records are dropped or added incorrectly or are otherwise corrupted, the problem may be quickly determined during a backup operation, since the control record will not agree with the records it represents. Another, non-exclusive, method used is that of the matrix control totals (sometimes known as cross-footing). This method operates in a similar mode as the hierarchial method, but uses double totals arranged to check record counts and totals from at least two directions. It may be even more powerful than the hierarchial technique, sometimes pointing to individual records with critical errors.

Recovery Failures

Software, hardware, or human problems encountered during backup or recovery operations may very well be fatal to recent updates to the data base, or in some cases, to the entire data base. Temporary problems initially require a restart of the backup or recovery operation, and may require a different method to be used. Permanent problems may require the restoration of an older copy of the data base and manual recovery procedures. The depth of alternatives for backup and recovery is determined by the cleverness of the data base designers and by the economics involved in the criticality of the contents of the data base.

Fault-Tolerant Data Management System Design

In order to minimize the problems described above, certain design decisions may be made in the data management system. The concepts presented briefly in the next few paragraons are not intended to be all-inclusive but are intended to give an indication of the methods which have been used and to help the reader understand the problems and proposed solutions.

One technique used in many very large data management systems, such as credit-card billing systems, is the cycle or sub-data-base concept. The data base is divided into some number of smaller sub-dats-bases, usually five to twenty, but sometimes fifty to one hundred, based upon a primary key, such as account number. I/O service routines in the data management system automatically direct inquiries and updates to the proper sub-data-base. This separation is normally done in a manner natural to the operating system software and I/O access system, but usually involves the placement of one sub-data-base per file or other logical system unit. Then one or more of the sub-data-bases may be disabled without affecting the remainder of the data base. Rotating schedules for file backup may be implemented while the system is up (except for the sub-data-base being backed-up). In case of hardware failure, there is a high probability that only one sub-data-base is affected, in which case only that sub-data-base may be taken off-line, recovered, and be placed back on-line. Even if partial or manual recovery procedures are required, they will probably be restricted to only one sub-data-base, thus tremendously reducing the recovery effort. The division of the large data base into smaller ones may also be very convenient for off-line or ancillary processing, such as for billing operations. For example, if the data base were divided into twenty cycles, one cycle could be billed each working day of a month, usually leaving several spare days. This is, in fact, the method many large credit-card processing centers use and is the reason that the billing date cannot be changed without re-issuing the card with a different account number.

Another technique often employed for data bases with very few updates is the pseudo-update method. The data base is not updated on-line, but the updates are saved in another data base. When inquiries are made into the primary data base, the information from the primary data base is automatically updated with the information from the secondary data base, if any. Although this causes a slight time delay due to the extra I/O activity required to service an inquiry, the primary data base remains unmodified, with all of the changes concentrated into the secondary data base, which may also be used as an audit trail for offline updating and other processing.

Most data management systems use indices to relate external key values, such as account number, to internal information locations. The method of updating the indicies when the information in a data base is changed must be carefully planned, balancing efficient use of the computer system with fail-safe updating of the data base indices and information records. Several on-line data management systems in current use maintain the entire set of high-level indices in memory and update them on disk only when the on-line monitor is being shut down. Obviously, in case of an abnormal termination, the data base is left in an invalid state if any records have been added or deleted, since the indices to the records would not have been updated permanently on disk. Several other on-line data management systems in current use refresh the data base and indices on disk whenever records are added or deleted. This technique is slightly slower per update but is superior from the criteria of data base integrity. However, even this may not be sufficient to always ensure the agreement of the data base records versus the data base indices. In the case of several disk acceses to a data base, the order of the acceses must be planned to minimize the damage if the sequence is not completed. Thus information records should be written before index records pointing to them are written, index records should be written before header records, etc. However, when using higher-level I/O access systems, the order of disk accesses may not be under program control. From a survey of several major I/O access systems currently in use on several types of computer systems, from micro to mainframe, the I/O systems seem to be designed for maximum, not minimum, damage to the data base in case of failure during the addition or deletion of a record. A compromise to the constant updating of the indices to a data base, which is independent of the order problem just discussed, involves updating the indices every "n" accesses or "s" seconds, whichever comes first. This method is also used by several data base management systems in current use.

Since humans are an extremely important element of any data base management system, they must be considered very carefully in the design and implementation of the data base, applications, and backup and recovery systems. Written instructions on the operation of the computer software and hardware should be provided, as well as personal instruction, in case of complex or new procedures. The maintenance of a manual audit trail for an appropriate amount of time after the data base updates have been completed is sometimes ignored but may prove of tremendous value in some of the situations described above.

Summary

This discussion has presented some of the major concepts involved in the design of backup and recovery oriented data managememt systems. The primary point is that the backup and recovery procedures must be integrated into the design and operation of a data management system for maximum functionality, effectiveness, and preservation of data integrity. The design of the backup and recovery system, like that of the data management system overall, is governed economically by all the local tradeoffs and priorities related to the management of the system.

DEBUG MOTO D2

DEBUGGING THE MEK6800D2 MICROCOMPUTER KIT

Lloyd Maul

Motorola Semiconductor Products Sector

Continued from July

Basic Debug Procedures

Malfunctions that may have been detected from the preceding operations will obviously need to be checked out. If it is a case of a malfunctioning key, the solder connections of the malfunctioning unit should be inspected. It may be checked with an ohmmeter to verify contact closure when the key is depressed. A few keys have been found that do not return to their original position after depression. Many times this may be fixed by removing the key cap and spring retaining washer. The return spring may now be stretched to apply a little more return force and then reassemble. If a switch problem is not cured by one of these actions, then a replacement switch should be obtained.

The most common malfunction at start up is the failure to achieve the dash prompt sign after a momentary depression of the reset button. A malfunction indication that has been seen is where the displays dimly indicate all 8's except the first one, which may have a zero in it. This indication is a clue that the wrong transistors for Q_1 through Q_7 on the I/O board have been packaged in the kit and installed. They should be MPS2907 units, while packaging errors have supplied MPS2222 devices in a few kits. The 2907 devices are PNP's, while the 2222's are NPN's and basically an inverted display is obtained. Another malfunction indication is where there is no display at all. This indicates that the system has failed to properly execute its initialization program and may be due to a variety of reasons. Many times this can be traced down in a methodical fashion as indicated in the following material. It is normally desirable to utilize an oscilloscope, although some valuable information can be obtained with a multimeter set on a low dc voltage scale that adequately covers a five volt range. Care must be taken with the interpretation of a multimeter reading. For example, a node that should be exhibiting a steady positive logic one or high condition may indicate a voltage lower than expected because of loading caused by the multimeter on an MOS semiconductor device. The phase one and phase two clock leads, when properly operating, will read something below 2.5 Volts because of loading and the averaging effect of the meter on a cyclic waveshape. The following discussion assumes the use of an oscilloscope. Device numbers referenced correspond to those on the schematics supplied with the kit.

An initial check should be made of the signals on the individual leads of the MC6800 processor with the reset switch held in the depressed state. This check can be quite revealing of possible improper connections.

Table I indicates the signals that should be observed in this quasi-stable situation. In many configurations other than the D2 kit, the data bus will display the data word indicating the most significant byte of the restart address. This is because the address of that byte responds to the present state of the address bus (hexadecimal FFFE) and the read/write signal is high effecting a read operation. On the D2 kit, however, selection of the ROM with the restart bytes is inhibited because the Valid memory address (VMA) signal is low and is used in the decode process by U11. Without a device being addressed, the D2 kit data lines all indicate a logic zero or low level. Under this reset condition possible shorts of the address leads to ground (except A_0) will show up and possible shorts to V_{cc} or an address lead will show up on the data lines. A complete check of the processor pins will verify proper operation of the clock and processor control lead states.

If observation of the signals on the MPU pins does not provide a clue to faulty operation, the signals in Table I may be utilized for the corresponding pins on the J-Bug ROM (U8), the J-Bug scratch pad RAM (U13), and the interface PIA (U21). This type of check should be made with the probe right on the device pin to check for signal continuity through the solder connections.

If the preceding technique of signal observation in the reset mode does not isolate the problem, waveshapes may be observed on the various pins in the operating state (reset switch normal). The system should be in a cycling mode. It should be noted that the waveshapes may not be as clean as might be expected because of signal crosstalk and scope lead ground loop pickup, however, the logic levels are usually within the specifications at the time they should be. This type of observation can pick up possible shorts between two address lines or two data lines that would not have shown up in the quasi-static observations. This check may also reveal a counter type operation on the address bus, that is the A_1 lead will have a fifty percent duty cycle waveshape that is half the frequency of the A_0 waveshape, A_2 will be half the frequency of A_1, etc. This will occur if the processor has read and executed an undefined operation code e.g., a 9D. Execution of such an op code can put the processor into this cyclic mode and only a restart will allow it to recover. Such operation is indicative of an improper address and/or data path between the ROM and the MPU and these leads should be examined and tested again.

If signals appear satisfactory on the microcomputer board, the problem may reside on the I/O board or the cabling between the two boards. The display as well as key closure detection is dependent on periodic low levels on pins 3 and 5 of devices U7, U8, and U9 on the I/O board as indicated in Figure 2. Pins 1 and 6 of these devices must be at a high level. Absence of proper signals should be traced back through the inputs of these devices (pins 2 and 7) and the cable to pins 10 through 15 on the interface PIA (U21) on the microcomputer board. The collector of Q_1 on the I/O board should have periodic high levels on it while the other translators (Q2 through Q7) should remain low. A malfunction of an individual display digit may usually be traced with these signals to determine improper connections or perhaps a faulty display device.

Signals on the data selector (U10) on the I/O board may indicate possible problem areas when trying to key in information or commands. The signal levels appearing on pins 3,4,5, and 6 are sequentially gated to pin 7 under combinational control of the signals on pins 2 and 14. Thus a low level placed on one of the input pins by a key closure should be traced to pin 7 and back through the cable to pin 9 of the interface PIA (U21).

Cassette Interface Debugging

Once the basic microcomputer system is up and running, much can be learned about microcomputer instructions and operation by keying in short program routines, tracing the individual instructions, and observing results by register examination. There comes a time, however, when a program is keyed in and it is desirable to store it on an audio cassette so that it can be later loaded back in without the necessity of rekeying the program. This is the function of the P(punch) and L(load) keys. When properly operating, a 25 millivolt peak to peak 1200 and 2400HZ (space and mark respectively) signal should be observed at the microphone terminal when the punch routine is set up and executed. This signal when recorded on a cassette recorder may be played back audibly to verify a recorded signal. When in the load function, the signal applied to the cassette earplug connector should be set up with the volume control to have a 3.5 - 5 Volt peak to peak signal for proper operation. If the recorder is equipped with a tone control, it should be set such that the level of the 1200 and 2400HZ frequencies are of equal magnitude.

The easiest way to verify proper operation of this portion of the circuitry is to punch a simple program that has been keyed into memory onto a cassette. Momentarily turning off the power supply to the kit will destroy the original program in the kit memory, but it may be reloaded by use of the L key after the cassette has been rewound and the system restarted. Proper operation is indicated by the reappearance of the dash prompt display after the program has been loaded, which follows a recorded 10 second all data marks (logic 1's) preamble that precedes the actual program on the tape. Existence of the original program reloaded into the same memory addresses may be verified with the memory examine function. Absence of the dash monitor after an appropriate period of time obviously indicates a malfunction.

The 30 second preamble of all data marks stored on the cassette is a good signal to help in the debug process. Even though the data is all marks, the signal has a single space for each data word and signifies the start of that data word. The most common symptom of a malfunction in the cassette interface is the failure to establish the proper clock signal to gate data into the ACIA (U23) on the microcomputer board. The proper clock can be detected by setting up the oscilloscope to sync on the receive data signal on pin 2 of the ACIA, while executing the load function with the 10 second preamble. The single negative going space transition in a word can be used to externally sync the scope and observe a complete word time frame. Once the sync is established, a rewind of the cassette tape and repeat of the load function will allow looking at the receive data clock signal on the ACIA (pin 3) to verify that a positive going edge occurs very near the middle of the single space bit interval (See Figure 3). A dual trace scope is very handy here, however, it is possible to check this using a single trace scope with external sync as described. If the proper data and receive clock waveforms are not observed, then a tracing of the clock and data signal paths through the cable and devices U17, U11, U18, U20, U13, and U14 may be necessary. Typical waveshapes that should be observed through this section are shown in Figure 4.

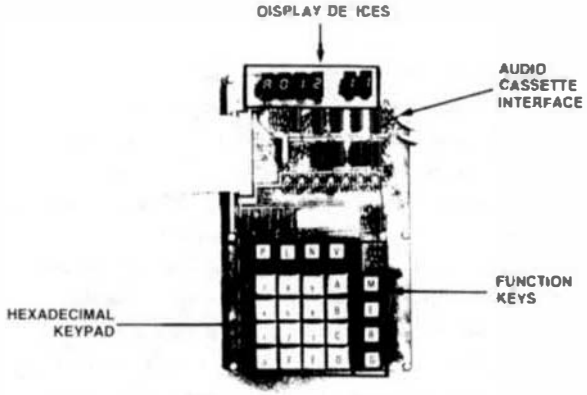
If it is determined that proper recording signals are not present during the record function, examination of the signals during the punch operation on the multiplexer (U20) may be revealing. The data levels coming out of the ACIA transmit data pin (U23, pin 6) should appear on pin 11 of U20. A 1200HZ signal should be observed on pin 12 and a 2400HZ signal should be appearing on pin 13 of U20. Failure to observe these tone signals may indicate the absence of a transmit clock (4800HZ) on pin 1 of U19 or pins 2 and 8 of U13. The RTS signal should be high (pin 10 of U20) in the punch

mode to enable signaling out of pin 15 of U20. Observation of these signals should isolate a punch malfunction.

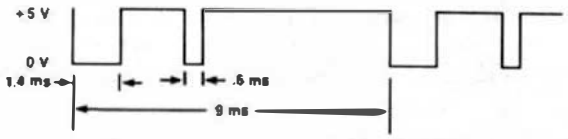
Conclusion

It should be reemphasized that most kit malfunctions have been found to be caused by improper solder connections with the major job being a case of determining the location of the bad connections. There have been instances where wrong parts have been included in the kit. This has most often been transistors, which can be easily overlooked during the construction phase. Only in rare instances have defective parts been a problem, but this should not be ruled out in the debug process.

The need to apply debug procedures to get a kit up and running can be irritating. However, the debug process can provide a lot of insight into the actual operation of a kit, which can be a very educational experience of its own. The attainment of a complete proper functioning system will provide the user with an extremely valuable learning tool plus the potential of a stand alone microcomputer for his unique end use application. Results beyond this point are only limited by the ingenuity and experience of the user.



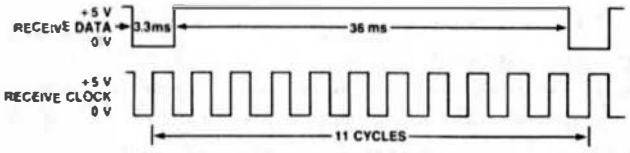
INPUT/OUTPUT MODULE
FIGURE 1B



NOTE: LOCATION TIMEWISE OF THE NARROW PULSE IS DEPENDENT ON THE PIN BEING OBSERVED.

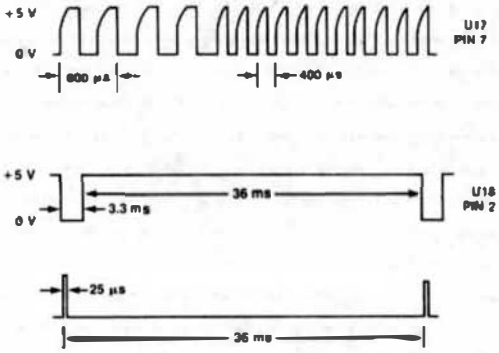
TYPICAL WAVESHAPE ON PINS 3 OR 5 OF U7, U8 AND U9

FIGURE 2



RECEIVE DATA AND CLOCK WAVESHAPES

FIGURE 3



TYPICAL CASSETTE LOAD WAVESHAPES

FIGURE 4

TABLE I

PROPER MC6800 SIGNALS WITH THE RESET BUTTON DEPRESSED

PIN NUMBER	DESIGNATION	SIGNAL
1	V_{SS}	GND
2	$\overline{HALT}$	+5V
3	ø1	50% duty cycle, 614.4K HZ, approx. 5V
4	$\overline{IRQ}$	+5V
5	VMA	GND
6	$\overline{NMI}$	+5V
7	BA	GND
8	V_{CC}	+5V
9	A_0	GND
10	A_1	+5V
11	A_2	+5V
12	A_3	+5V
13	A_4	+5V
14	A_5	+5V
15	A_6	+5V
16	A_7	+5V
17	A_8	+5V
18	A_9	+5V
19	A_{10}	+5V
20	A_{11}	+5V
21	V_{SS}	GND
22	A_{12}	+5V
23	A_{13}	+5V
24	A_{14}	+5V
25	A_{15}	+5V
26	D_7	GND
27	D_6	GND
28	D_5	GND
29	D_4	GND
30	D_3	GND
31	D_2	GND
32	D_1	GND
33	D_0	GND
34	R/W	+5V
35	No Connection	GND
36	DBE	50% duty cycle, 614.4K HZ, approx. 5V excursion
37	ø2	" " " " " " "

J8	No Connection	GND
J9	TSC	GND
40	$\overline{RESET}$	GND

Low Cost Winchester

David J. Graves
311A Towne Building/D3
University of Pennsylvania
Philadelphia, Pa. 19104
(215) 898-7951

Some time ago I had decided that those two minifloppy drives which seemed so marvelous 5 years ago just weren't adequate any more. With less than 100K of storage each, and FLEX plus a few utilities filling up one of them, I found that I just couldn't work effectively on a moderately large program. However, I was undecided what to do. With double-sided, double-density, and double-track options available (not to mention the added choices if 8" rather than 5" drives were chosen) I could get up to about 0.8M per new drive, but such drives are not particularly cheap, and a new controller board would add several hundred dollars to the total cost. A Winchester hard disk seemed the better solution in storage per dollar invested. Unfortunately, the entry price into the hard disk market until very recently has been quite steep.

When I saw a 5M drive advertised for $668, with a companion intelligent controller for $588, I knew that this was the answer (the vendor, Computer Dynamics, has an ad elsewhere in this issue). A commercial power supply, connectors, cables, etc. added another $200-300 to the total price. I had an old tape recorder case which could neatly house the drive, controller, power supply, and a fan (absolutely essential!). I now have a system with the two minifloppies (drives 0 and 1) and the 5M Winchester configured as drives 2 and 3. The hard disk could have been configured as a single drive, but the 2-drive arrangement permits backing up files and generally is more convenient. In any case, I will have no need to add another drive for quite some time. If and when I do, a simple re-write of the driver program can convert it to a single drive. In the remainder of this and a subsequent note, I will give a mini-tutorial covering a few facts about Winchesters, FLEX, and the hardware and software needed to interface the two. Although I would not suggest this as a project for a complete neophyte, the overall job is not particularly difficult for someone with moderate manual skills and some experience with hardware and software projects.

To begin with, FLEX is designed to handle up to 4 drives, each of which can have up to 256 tracks, with 256 sectors (of 256 bytes each) per track. This turns out to be 16M bytes per drive. The original minifloppies have only 10 sectors per track and 35 tracks, so with these limits TSC left plenty of room for expanded capacity. Winchesters have one or more stiff and very smooth metal discs coated with magnetic material and permanently sealed in a very clean environment. These and other design features such as special low-flying read/write heads allow high recording densities. In fact, the recording densities seem to be continually increasing, which results in smaller discs, higher capacities, or both as manufacturers come out with new models. The current most commonly used drive has 5" discs or platters, each disc having a R/W head on both top and bottom surfaces and holding 5 or 6M bytes. The drive on which this note is based (a Shugart SA 1002) is an 8" drive with the same capacity. Being somewhat older, it is available at a lower price than the 5" drives, but one pays for this in the form of a larger size, higher power consumption, and higher heat production. At the other extreme, a 3.9" Winchester has been announced recently by Sy Quest Corp, which holds 6.38M on removable cartridges. These cartridges, with a self-closing door, combine the best features of the floppy (multiple copies for backup) and the Winchester (high capacity), and may eventually revolutionize the mass-storage scene. The hobbyist will not find the latest technology like this micro-Winchester at bargain basement prices, but it has been predicted that the fierce competition in this field will result in significant price reductions in 5" drives within the next year. Even without additional price reductions, though, the Winchester should be seriously considered for advanced personal computer systems even today.

One concept which may not be familiar to the floppy disc user is that of the "cylinder". All R/W heads (2 heads on a 1 disc drive, 4 on a 2 disc model, etc.) move in and out simultaneously. A single physical location for the heads is called a cylinder and contains all of the tracks on the top and bottom of the disc(s) at that location. The sectors in all tracks can then be concatenated to form a larger number of sectors in the cylinder. For example, the SA 1002 can support 32 sectors of 256 bytes each per track. The top of the disc could be called sectors 1-32 and the bottom called sectors 33-64 if one wished to configure the drive as a single entity. This strategy reduces the amount of head movement needed to read or write a file. A two platter drive such as the SA 1004 could have 128 sectors per cylinder. (In my case, configured as one drive per side, the cylinder number can be considered equivalent to the track number in the following discussion, and for the SA 1002 can range from 0 to 255.

The Western Digital controller card, the WD 1000, contains a fast microprocessor which makes interfacing quite easy. It can transfer up to 4.34M bytes/sec, control up to 4 drives, automatically generate and check CRC codes, and if an error is detected, automatically read or write the data again. On read and write sector commands, the controller automatically sends commands to the drive to seek the proper cylinder. A particularly valuable feature of the controller is that it can also handle the Quantum Q 2000 and Seagate Technology ST 506 drives with at most a few parts and wiring changes. The Seagate is a 5" drive, and its control interface has become somewhat of a de facto standard, so a number of other drives should be compatible as well. Furthermore, Western Digital has recently announced the WD 1001, which is promised to cost less and provide error correcting features, the WD 1002, which will handle both floppies and Winchesters, and 2 other controller boards as well. There is thus reason to hope that the information presented here should retain some value for quite a while.

CONSTRUCTING A SYSTEM

Now, on to the details. In addition to the drive and controller, one needs an appropriate power supply. I chose model CP 384-A by Power One. If the A suffix is omitted, one can save a few dollars by sacrificing 120/240 V compatibility and electrical insulation satisfying European safety standards. Two 50 contact and one 20 contact card edge connectors are needed, in addition to one 50 contact, one 40 contact, and one 20 contact header sockets (for dual rows of pins on 0.1" centers). The 40 contact socket will mate with a vertical header strip (which should also be purchased) on the 6809 interface board. I strongly advise getting connectors which can be squeezed in a vise to automatically pierce the insulation of ribbon cables and make contact with all wires simultaneously. Making that many solder connections would be a horrendous task, and the reliability of connection with the crimp-on sockets is excellent. A continuity check is still, of course, necessary. Three feet of 40 conductor ribbon cable, and enough 50 conductor and 20 conductor cable to connect the drive and controller are also needed. The latter two can be up to 10 feet in length if one chooses to separate the controller from the drive. The computer and controller, however, must be separated by no more than 3 feet of cable. Several power cable connectors also are required, but if you cannot obtain them, you can improvise as I did or solder directly to the appropriate points. In mounting the drive and other components, you must be careful to provide adequate air flow. A good fraction of the total heat is generated by the AC drive motor. The fins built into the motor and the adjacent slots in the cover of the SA 1002 do not provide much help in removing it.

The interface to a 6809 system is fairly straightforward. A version for the 6800 AND FLEX 2 should not be particularly difficult. The MC6875 clock generator has a "memory ready" input, and 6800 systems using this chip should be able to add a wire to the MRDY bus line (after connecting the reset button directly to the CPU card and cutting free the trace to what used to be called MRESET). I have not yet tried to do this, however, so there may be a few problems that I am unaware of. An improved version of the wire wrap prototype I will describe here is being committed to a printed circuit design and should be available by the time this appears in print. (Drop me a line if you are interested). I also have a Color Computer. If enough 68C fans are interested in a Winchester, I might be persuaded to work out the details for this system too. Basically, the WD 1000 requires 3 address lines for the eight registers it contains, 8 data lines, board select, read and write signals, itself. This is an area which obviously should and will be improved. Another obvious area for improvement is to use only 74LS devices, rather than the mixture of regular and LS TTL which my junkbox provided. The purpose of the flip-flop is to cut short the write pulse as soon as the WD 1000 is ready to accept data, signalled by MRDY going high, rather than after the 6809 WRITE signal goes high. I found this re-synchronization essential in my system. I have not tested this interface with a 2MHz processor, but see no reason why it should not work. The diode prevents MRDY from halting the system if one wishes to operate with the Winchester turned off. (In the circuit diagram, oval-boxed symbols refer to the SS-50 bus signals. Bullet-shaped boxes enclose numbers referring to the WD 1000 host connector. Names are also given for those signals which are not obvious. All even-numbered controller lines (not shown) should be grounded.

The use of MRDY to temporarily stop the 6809 processor by clock stretching is essential. The WD 1000 will occasionally take up to 6 microseconds to accept or deliver data, depending on what phase of its operations is taking place when a command arrives. In my case, none of the other parts of my system used MRDY, and I was unaware of a problem (mentioned to me by Mr. Zeff) which at first prevented it from functioning properly. Four early production batches of 6809 chips (lot numbers G7P, T5A, P6P, and T6M) do not properly stretch the E and Q signals. If your CPU has any of these lot numbers stamped on it, as mine did, the easiest solution is simply to replace the 6809 with a new one. Motorola outlines a "fix" in the 6809 information bulletin, but it requires the addition of extra components to the CPU card and probably is not worth the trouble now that the CPU chip itself is less than $15.

The only difficulty I had in getting the system operational originated from an incorrect ROM in the controller board. Although the WD 1000 manual discusses commands which select sector size, these do not function unless the correct ROM is installed. This problem is now mentioned in a sheet included by Computer Dynamics, but their WD 1000 boards come with a 512 byte sector ROM as standard unless you specify the 256 byte sector ROM. I would suggest contacting them directly if you wish to use the controller with a drive other than the SA 1002 in order to make use of my interface and software, because of the different parts and wiring which might be needed in such a case.

Assuming that everything has been properly connected (and at least triple-checked), the first test would be to use SBUGE

or other monitor to read locations SE030-SE037. Location SE032 should contain SFE, location SE037 should be SAF, and all others should be SFF. If this test is satisfactory, store 08 in SE034 and then S8F in SE037. The drive should emit a high-pitched "beep" as the stepper motor rapidly moves the heads to the reset, and a wait line connected to MRDY. I have chosen to incorporate two 2016 static RAM chips (2K x 8 each, but one not fully used) on the interface board as well, adding 3.75K of memory to my system. The disk driver software resides in almost 2K of this memory. By locating the memory in the SE100-SE7FF area, I have "hidden" it from programs which search for the end of contiguous memory, and it will not interfere with these or any other standard programs. SBUGE does not see it either and reports only 56K rather than 59.75K of memory in the system.

I have chosen to put the interface circuitry and this memory on a 50 pin card rather than use a 30 pin I/O card with the interface addressed at SE030-SE03F. This is beyond the output port range in my system, which has 8 ports at 4 addresses each from $E000 to SE01F. On newer systems with 16 addresses per port, SE090-SE09F might be a better choice for the interface address. Two wires can be exchanged and the circuitry should work at the higher addresses. (Exchange A7 and A5 to the decoder gates but not to the memory).

It goes without saying that the motherboard must completely decode output port addresses in order to implement this scheme. Alexander has published a method to do this on older MP-8 and MP-B2 motherboards by SWTPC (see April 1980 68 Micro Journal, p. 23. A number of other motherboards, such as those by Cimix, Helix, Thomes Instrumentation, and Data Systems 68 appear to be fully decoded and should require no modifications. My interface uses up 16 addresses rather than 8 because the A3 address bit is not decoded. Note that only 1.75K of the lower 2016 RAM is used, which leaves 256 addresses for output ports. Since even the new 16 address per port system uses only 128 addresses, this should be adequate for virtually all systems. A 2716 EPROM could be substituted for either of the 2016's by tying pin 21 of U7 or U8 to 5 volts. Since I have switch-selectable 6888 and 6809 CPU's, I have added the relay shown so that all of these addresses can be in the $8000 range rather then SE000. If this feature is not needed, the relay and associated parts can be omitted. Many CPU cards permit inserting a 2K X 8 RAM chip from SE800-SEFFF, so the extra memory would not be needed by everyone building an interface from scratch. The only thing sacrificed would be the ability to move the memory address along with the output port address if it were switched.

A 74LS245 is used as the data bus driver. This was convenient since it transmits inverted SS50 bus data to the WD 1000. I have modeled my software routines after those written by Robert Zeff, and these contain commands in the inverted form. If this inverted data is esthetically unpleasing to some, a 74LS640 may be substituted (it is pin for pin compatible with the 74LS245) and all the software rewritten with the commands and bit tests replaced by their complements. Note that while all signals between the computer and the WD 1000 are buffered, I have not buffered all of the address lines to the interface innermost cylinder. An $FF in SE034 and another S8F in SE037 should cause another beep and a return to the outermost cylinder. This action can be seen through the plastic cover of the SA 1002 and is a rather startling change from the slow track-seeking common with some floppies. In each case, re-examining location SE037 will show that it contains SAF, signalling successful completion of a command, rather than the S8F that you stored there. Location SE034 is the cylinder register and SE037 the command register, and you have just issued (the complement of) commands to seek the highest and lowest cylinder numbers by this sequence. A return to-cylinder-zero command can also be given simply by issuing a "restore" command (put an SEF in SE037). Three of the registers, 0,1, and 7, are "read only" and "write only". Do not expect to be able to store and retrieve data there as you can with normal memory locations. If everything works as indicated, there is a good chance that no mistakes have been made and that you are ready for the software which will be published next month. The final result of these labors will be a system which reports 8160 free sectors when a CAT 2 or a CAT 3 command is issued in FLEX rather than the 340 sectors you have been accustomed to with minifloppies.

U1 - 7427

U2 - 7400

U3 - 7420

U4 - 7404

U5 - 74LS138

U6 - 74LS245

U7,U8 - 2016 (8118)

U9 - 7486

U10 - 7474

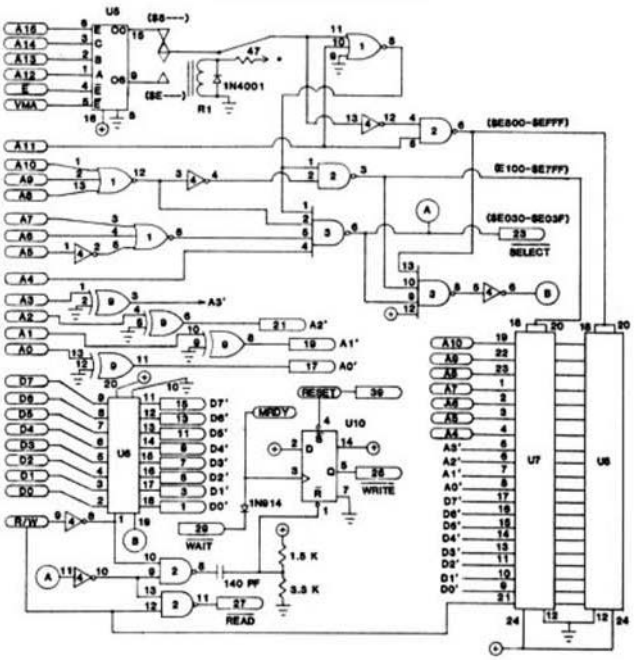
R1 - RADIO SHACK 5 V. "SUGAR CUBE" RELAY

U1 - U4 & U9 POWER (PIN 14) AND GROUND (PIN 7) NOT SHOWN

*** - TO FRONT PANEL SWITCH (8 V.)**

WD 1000 TO SS 50 HOST INTERFACE

PLUS 3.75 K MEMORY



DMS NOTES

Bill Adams
21 Larch Road
Briarcliff, N.Y. 10510

The term "Data Management System", or "DMS" refers to a set of programs designed to facilitate the implementation of generic applications. This loose definition is intentional since DMS's typically vary in capacity and functionality. Generally a DMS can be thought of as something between an operating system and a user application, both in level and in communications between these entities.

In this writing, it is my intent to explore some of the rationale and design of DMS's, with particular emphasis on DMS2/VM, which I wrote for Westchester Applied Business Systems. In preparing for this article, I soon discovered that it would take reams of paper to explore all facets, and that time and print space would be constraints. I will attempt, therefore, to concentrate only on certain important points, and willfully omit others.

Before jumping directly into DMS architecture, it is necessary to define some terms. The subject of this article - Data Management - suggests two main catagories, which, appropriately should be understood. First, the data portion. Most DMS's employ "field" to mean an indivisible piece of data, usually of fixed length, to which a name or

label is given. Fields usually have attributes as to the type of data they contain, such as "numeric" or "alphanumeric". If we classified the Basic language as a DMS (a very loose analogy) we would find string and numeric variables available with single letter "name" values. DMS's normally provide a wider range of field types to accomodate various data formats. Field names are usually large enough to allow for meaningful identification by the user. DMS2/VM for example provides for alphanumeric, numeric, integer, decimal, coded, hexadecimal and scalar date field types with up to eight character field names.

The "field", which can then be thought of as a fixed-size, variable piece of data, is the building block of data management systems. By itself, however, it is quite useless and must be related to other fields in some way. A telephone number, for example, is associated with a name. We therefore group fields together by placing them adjacent to form a "group" or a "record". (The term "group" under DMS2/VM implies hierarchical organization and will be discussed later.) A "record" may be thought of as a line of data which has been formatted into fields. If we were to type a record with a typewriter, we would first set the tabulation mechanism, then type each field (justified left or right) after positioning to the corresponding tabset. In this manner, multiple records of the same format, when typed, form vertical columns and result in an ordered, easy to read document.

A collection of records is referred to as a "file". Our paper version of a file produced by the typewriter, above, is closely related to the internal structure of a DMS file, such as that stored in memory or on disk. Memory and disk may be thought of as an array of "bytes" which may contain letters and numbers, and like our typewriter page, may be "tabulated" or "formatted" into fixed length fields, records and files. Since internal bytes can represent up to 256 values, certain field types such as numerics can be compressed into a lesser field size, so that the overall internal record and file sizes are reduced. This is particularly desirable on small systems, since it permits larger files to exist on limited physical resources, eg; floppy disk. DMS2/VM uses 2-for-1 compression on all but alphanumeric fields.

We skipped over "group" previously. Under DMS2/VM, it refers to a collection of adjacent fields which are logically separate from another collection of fields. Fields, thus, belong to one group or another. Suppose for a moment that we named the columns of our typewriter output as NAME, SSNO, WEEK, MON, TUE,...SAT with the intent of recording hours worked for a number of employees. Furthur, we will also keep our records (sorted) by NAME and WEEK so that missing or duplicated records can be easily identified. If this is the case, we will find that NAME and SSNO become redundant within adjacent records (52 occurences per year) while the weekly data, (WEEK, MON - SAT) is not. DMS2/VM allows us to define these fields within separate "groups", which are treated as subrecords of the main record, and which are suppressed if redundant. This scheme establishes levels of dependency, where the leftmost group is considered level 01 and the rightmost 02. Lower levels (02) are said to be dependant on higher levels (01).

While the logical view of the file is still one of tabular form, the physical contents consists of a series of groups, ordered as 01,02,02...01,02... Properly applied, this hierarchical structure can substantially decrease storage space required, and permit larger (effective) files on existing media. DMS2/VM provides for up to 12 group levels and automatically suppresses redundancy for all but the lowest level group.

One last definition in the data area. "Database" has been used to refer to anything from a sequential file to a hierarchical structure (as above) to a network of files interrelated by pointers or keys. Generically, it refers to the collection of all data for an application, somehow organized to be accessible. Normally, the relationship of data within a database is governed by the functionality of the DMS, which leads us into our second set of definitions; Those for the management of data.

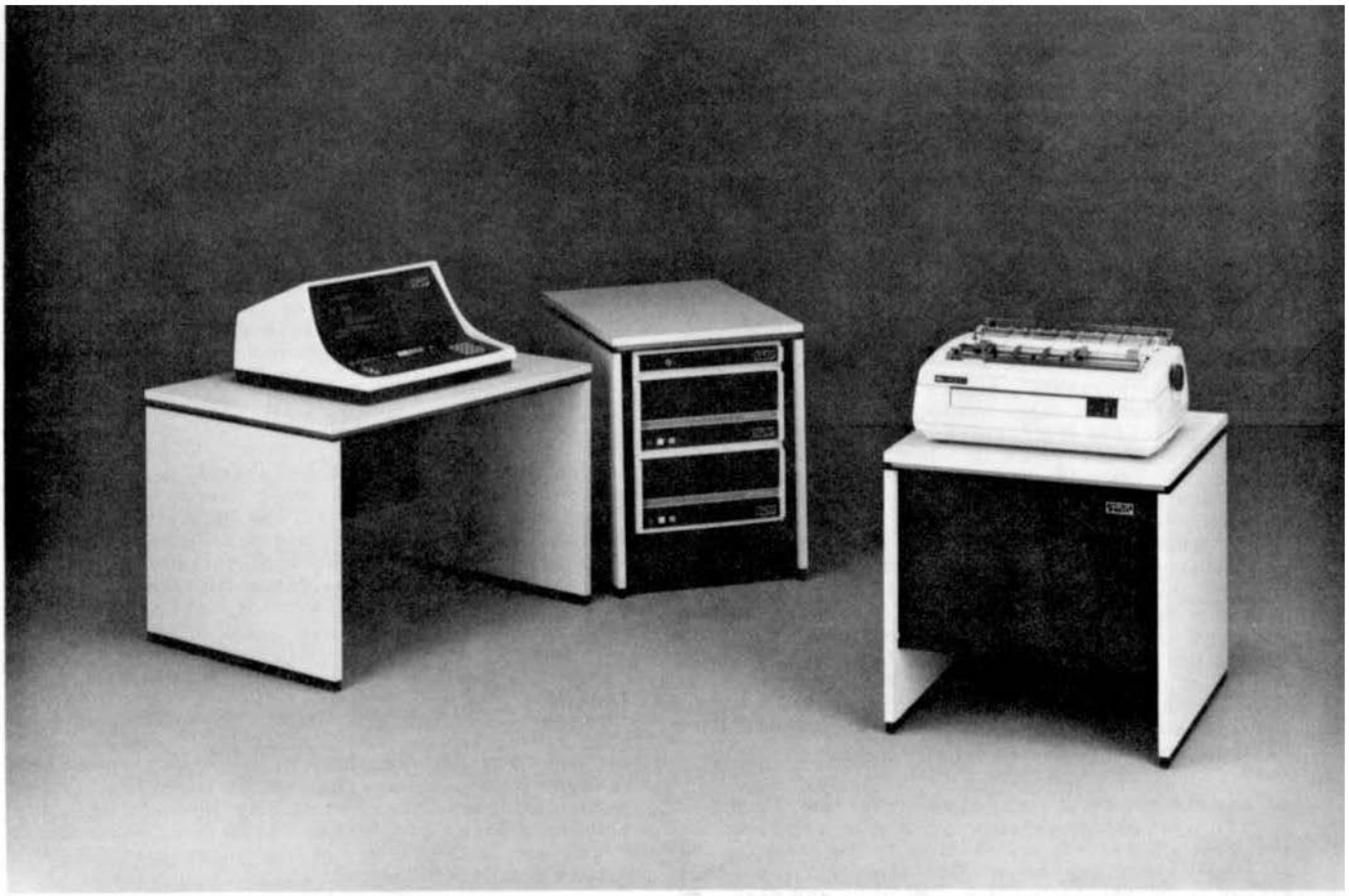
In the context of top-down design, we have just defined the first level of a hierarchy. This is sometimes referred to as the 'what' portion as opposed to the 'how-to' portion. In essence, we have also described many applications, which for our purpose is quite desirable. But now our task is to further define these design components into more discrete functions.

I will not elaborate on the file definition or update functions of DMS systems, but rather consider them as "trivial". They are in fact simple programs within the DMS2/VM system which provide editing capability on the format and content of a database file. More important is the manipulative capability of a DMS in terms of being able to process and format the entered data. Here, I will not attempt to develop rationale, but will rather report on findings.

Normally the processing done by a DMS consists of a series of predefined generic functions, which may or may not be done based upon user control. These functions typically include the filtering of input (record selection), sorting, lookups, arithmetic, accumulation and totaling, and the formatting of output. The output format may contain only fields of interest to the user (field selection) and may be directed to the terminal, printer, disk spool file or another database file for further processing. These basic capabilities fulfill the needs of most "list oriented" applications.

The control of these functions can take several forms. A popular means of providing it is through a "non-procedural language". Such a language consists of statements which specify the format, content and order of the output without regards to the processing necessary. Non-procedural languages are intended primarily for those who have limited programming experience, but are also quite usable to preclude the necessity of writing a program. Generally, much can be accomplished with a small set of command statements.

Additional functionality is found in different DMS's and typically the 80-20 rule applies (80% of the effort goes into solving 20% of the problems). While a large number of applications fit into the DMS environment, many others are so diverse in their processing requirements so as to dictate specially written programs. In making the decision as to whether use a DMS, the user should weigh the necessity of functions which are not provided with the time and cost of developing specific software. Very often, a DMS will serve the purpose if minor adjustments are made to the User's specifications. DMS's also have the advantage of providing for quick reports, inquiry and other processing



THE COMPLETE BUSINESS SYSTEM

+Multiuser+Highly Expandable+Cost Effective

S+ THE CONCEPT

The S+ system is a modular computer system in which all portions of the hardware and software are designed to work together in the most efficient way possible. An S+ single user system with floppy disk storage is a competitive and cost effective entry level system. Unlike most other small computers being sold as "personal", or "small business" machines, the S+ system may be expanded to maximum capabilities using this same hardware and software. You cannot end up with a DEAD END system that cannot be expanded and whose software is not compatible with larger machines. A basic S+ system may be expanded to thirty-two users, a megabyte of main memory and hundreds of megabytes of hard disk storage by simply plugging in, or connecting the desired upgrade equipment.

TOTAL DESIGN—Hardware and Software

The S+ system is an integrated hardware and software design. The two complement and enhance each other in this system. The UniFLEX® operating system used in the S+ systems is patterned after the Bell Laboratories UNIX® operating system, one of the most admired and widely used operating systems in the world. Instead of being an afterthought, the software is part of the design of the S+ system. You can be sure that with this approach that all parts of the computer operate with maximum efficiency and cost effectiveness.

THE CENTRAL PROCESSOR

The basic S+ system is configured with 256K bytes of memory and can be expanded to more than 1 million bytes. An efficient and fast hardware memory management system is used to allocate the available memory among the users on a dynamic basis. As little as 8K bytes, or the entire memory—if needed—can be used by any individual user. This makes it possible to run very large programs on the system, but it also uses no more memory than necessary for a particular job. The increase in cost effectiveness of this system over crude and outdated bank switching arrangements is dramatic.

The central processor runs in both user and supervisor states. It can detect and reject a defective user program. It is impossible for a user program to go bad and stop the entire system, as can happen quite easily in less sophisticated systems.

Task switching is accomplished by use of a multiple map RAM memory, with sixty-four individual task maps. Each task can access from 4 to 64 K-bytes of memory. Multiple tasks may be used in programs that require more than 64K bytes of memory for execution. When a task is completed the memory is automatically released for other use.

SOFTWARE

The S+ operating system, UniFLEX® is a multiuser, multitasking operating system based on the UNIX® operating system that has been used for many years on Digital Equipment Corp. PDP-11 series minicomputers. It is considered one of the most sophisticated and "user friendly" operating systems available. Variations of UNIX® are rapidly becoming standard on mini and larger microcomputers.

A large variety of languages are available for use with the system. These include FORTRAN, COBOL, BASIC, and Pascal. Word processing packages are also available to give you full text processing capability on the system.

Applications programs are available in large quantities in many fields. This includes general business, medical, dental, veterinary, library and real estate management; plus others. Since the system is multiuser it can also be connected to cash registers to produce a point-ofsale terminal system combined with the computer. The possibilities for application of this system are endless.

THE I/O SYSTEM

The S+ system is totally interrupt driven. All terminal and printer I/O devices connect to an I/O bus separate from the main bus. Up to thirty-two separate devices may be connected to the I/O bus at any one time. If I/O activity is great enough to cause an unacceptable slowdown in system operation, a separate I/O processor can be installed in the system. This plug-in option removes all I/O handling overhead from the main processor and allows operation of up to thirty-two external devices at 9,600 baud. Without an integrated total design, as in the S+ system, it would become impractical to use a UNIX®type operating system in a situation with heavy terminal I/O activity.

DISK STORAGE

A wide range of disk storage capacity is available for the S+ system, from 2.5 M-byte floppy disks to an 80 M-byte Winchester and many sizes between. All disk controllers use direct memory access (DMA) type operations to maximize data transfer and to minimize overhead on the main processor. The Winchester disks also use intelligent controllers along with DMA transfers to preserve the performance that these type devices are capable of giving. Without this distributed intelligence the system performance would be greatly degraded. The UniFLEX®operating system is designed to work at maximum efficiency with this type disk system. The data transfer rates achieved by this combination rival those of large minicomputers.

COMMUNICATIONS

A high speed local network communications system is available to interconnect S+ systems. The VIABUS® network will allow communication between systems at data rates of over 400K baud. Such a system makes it possible to share data between local systems in an efficient and low-cost manner.

AVAILABLE SOON

Tape backup—20M-Byte in less than 15 minutes on a standard ¼ inch cartridge.

Mini-Wini—5 and 10 M-Byte Winchesters—5¼ inch package. Winchester performance, for smaller systems in a small package. UniFLEX® compatible design.

Large Capacity—190 and 340 M-Byte Winchesters, plus SMD cartridge drives.

UniFLEX is a registered trademark of Technical Systems Consultants, Inc.

UNIX is a registered trademark of Bell Labs.

VIABUS is a registered trademark of Southwest Technical Products Corporation.

functions which may not be originally envisioned when an application is developed.

When selecting a DMS, look for flexibility. Long term needs may dictate additional functionality. If your system has limited storage, look for data compression. (Multiply record size by estimated number of records for max. storage required - then compute with compression applied.) In some cases it may be found that additional storage is required. Purchase the user guide for prospective DMS's - Do not rely on advertising for assurance that it will work for you. You have several thousand dollars invested in your hardware - it is worth the $10. or so to insure that the software you select will do the job for you.

Next we will examine how several applications may be set up and run under DMS2/VM. This will serve to illustrate some of the above and expore some additional facets of DMS applications.

The programs which constitute a DMS provide the user with the operational control required to "manage" the data within a database. DMS designs seek to provide both flexibility and ease-of-use, although the two often conflict. This can be illustrated by a relative comparision with other program "entities" to exaggerate this effect.

ENTITY	FLEXIBILITY	EASE-OF-USE
Application Program	Quite Limited	Very Easy
Utility Program	Somewhat Limited	Quite Easy
DMS	Flexible	Easy
Basic Language	Quite Flexible	Not So Easy
Assembler	Very Flexible	Difficult

As implied here, DMS's seek to fill the middle ground between specific application programs and programming languages. Often, utility like functionality is found, as is a high-level control statement set. The functions available, and the control thereof, varies among data management systems.

Now that we have used the word in context, it is appropriate to define "function". This word became popular with the advent of structured programming, to describe a task that the computer was to do. No particular level of complexity is implied, since functions are typically composed of other (sub)functions which are in turn composed of other functions. This hierarchical organization of functionality has many benefits when properly applied. Although normally applied in a top-down approach, which decomposes an overall programming task into managable components, a combined "top-down, bottom-up" approach can often produce highly compact, efficient program code. This is because the low level functions (those closest to actual machine instructions) are examined with respect to their commonality of usage, and are modified to be generic. This allows the same function to be executed as part of many other functions, whereas the physical code occurs only once. DMS2/VM employs such a design approach in the "nucleus" portion of the code, which permits approximately 230 callable functions (subroutines) to exist in an 8K region.

But how does the user control all this? This is an important question. Control in a DMS environment relates back the the flexibility v.s. ease-of-use problem, and often results in a compromise. Perhaps the simplist form of control is a menu driven system, which prompts the user with multiple choices. A more complex, more flexible approach is to furnish the user with a set of commands which are interpreted by the system, and the appropriate functions executed. The most complex, most flexible approach is to allow direct user interface with the inherent functionality from a user written program.

As mentioned earlier, DMS refers to a set of programs designed to facilitate the implementation of a generic application. How, then do we define the architecture if we don't yet know what it is supposed to do? The answer to this question lies in the study of many computer applications, the identification of similarities and ultimately the definition of the functional bounds of the DMS, itself.

This analysis actually begins with the formulation of applications. There must be a way, for example, to define to the system the record and field formats used by an application. Once defined, there should also be a means to enter information, and also to modify or delete it if necessary. Once our database has been established, we wish to be able to retrieve the data and present it to our specifications. Oh yes, - We also wish to do all this with minimal effort (eg) number of keystrokes) - A stipulation which returns us to the flexibility v.s. ease-of-use problem.

BIT BUCKET

WINDRUSH
Micro Systems Ltd

Windrush Laboratories (Hing Cottage)
North Walsham, Norfolk NR28 0SA
Tel: (0692) 406188
Telex: 97380 SHARET G

68 MICRO JOURNAL
Attn: Tom Williams
5900 Cassandra Smith Rd.
Hixson
Tennessee
U.S.A. 37343

Your Ref. — Our Ref. WCD/pd 133 Date 19/8/82

Dear Tom,

As an endorsement for the advertising effectiveness of your publication we have sold over 500 of the 'ALL-IN-TWO' as a direct result of our advertisement in '68 Micro.....Yours is definately the best publication for reaching the 6800/6809 user WORLDWIDE!

Yours faithfully,

William C. Dickinson
DIRECTOR

68 Micro Journal
5900 Cassandra Smith Road
Hixson, TN 37343

May 14, 1982

Don,

Several people have asked for a 'Clear Screen' function for XBASIC running on the Color Computer. I thought you might publish this information for all FLEX Color Users in your Bit Bucket. It will perform the same function that a "CLS" in Microsoft Basic does. The next release of FLEX for the Color Computer will have this included.

A change must be added to the high-resolution video display routines to test for a 'clear screen' control code ($7F). This will allow a simple PRINT CHR$(127) in XBASIC to cause the entire video display to be cleared.

The attached partial source listing of one of the hi-res video display routines shows the additional code required. A similar change can be added to the 32x16 display. Make the source change, and re-assemble the routine with an appropriate binary filename.

Keep up the good work!

Steve Odneal
8609 East 73 Terrace
Kansas City, MO 64133

```
COLOR COMP. 42X24 VIDEO ROUTINES      4-8-82 TSC ASSEMBLER   PAGE   3
COPYRIGHT 1982 BY STEVE UHRIG

E42D B7  FFC5            STA   $FFC5     SET HIGH-RES
E430 39                  RTS
         E431  SETSAM    EQU   *
               * SET SAM TO $E000 VIDEO PAGE.
E431 B7  FFC7            STA   $FFC7
E434 B7  FFC9            STA   $FFC9
E437 B7  FFCA            STA   $FFCA
E43A B7  FFCC            STA   $FFCC
E43D B7  FFCE            STA   $FFCE
E440 B7  FFD1            STA   $FFD1
E443 B7  FFD3            STA   $FFD3
E446 39                  RTS
         E447  SCNCLR    EQU   *
E447 CC  FFFF            LDD   #$FFFF
E44A BE  E600            LDX   #$E0F0A0
E44D ED  01      LOOP1   STD   ,X++
E44F BC  FE00            CMPX  #$ENDPAG
E452 25  F9              BLO   LOOP1
               * RESET CURSOR TO TOP OF SCREEN
E454 CC  E600            LDD   #$E0F0A0  BEGINNING OF VIDEO PAGE
E457 FD  E40D            STD   CURSOR
E45A 39                  RTS

               *
               * ENTRY POINT FOR CHARACTER DISPLAY. BACK SPACE.
               * CARRAGE RETURN, AND BLINK CURSOR.
               *    REGISTER A = HEX VALUE OF CHARACTER (ASCII)
               *    $08 = BACK SPACE CURSOR
               *    $0D = CARRAGE RETURN) (NEW LINE)
               *    $FF = BLINK CURSOR (COMPLIMENT BITS)
         E45B  DISPLY    EQU   *
E45B BD  E42F            JSR   SETSAM    INSURE PROPER VIDEO PAGE
E45E 81  FF              CMPA  #BLINK    SHOULD WE DO A CURSOR ?
E460 27  20              BEQ   CBLINK    ..YES
E462 34  76              PSHS  A,B,X,Y,U SAVE EVERYTHING FOR FLEX
E464 81  08              CMPA  #BACKSP   BACK SPACE REQUEST?
E466 1027 0083           LBEQ  BACKUP    YES. DO BACKUP ROUTINE
E46A 81  0D              CMPA  #CRETURN  CARRAGE RETURN REQUEST?
E46C 27  3D              BEQ   SCROLL    YES. DO NEW LINE ROUTINE
E46E 81  20              CMPA  #$20      LESS THAN A SPACE ?
E470 2D  0D              BLT   RET2      ..YES. IGNORE IT
E472 17  0000            LBSR  DISPCH    ELSE. DISPLAY CHARACTER
         E475  RET1      EQU   *
E475 7C  E40D            INC   COL       UPDATE THE COLUMN NUMBER
E478 B6  E40D            LDA   COL       GET CURRENT COLUMN
E47B 81  20              CMPA  #32       PAST END OF LINE ?
E47D 27  2C              BEQ   SCROLL    ..YES. MUST SCROLL
         E47F  RET2      EQU   *
E47F 35  76              PULS  A,B,X,Y,U
E481 39                  RTS
               * CURSOR BLINK ROUTINE
```

CMPA #$7F
BEQ RET3

RET3 CLR COL
PSHS A,B,X
LBSR SCNCLR
PULS A,B,X
RTS

WINDRUSH
Micro Systems Ltd

Windrush Laboratories (Reg. Office)
North Walsham, Norfolk NR28 9SA
Tel: (0692) 406189
Telex: 97380 SHARET G

'68 MICRO JOURNAL
Attn: Tom Williams
5900 Cassandra Smith Road
Hixson
TN
U.S.A. 37343

Your Ref | Our Ref WCD/asd PL9/MACE | Date 20/8/82

Dear Tom,

Enclosed are the latest versions of PL9 and MACE as well as our ALL-IN-TWO EPROM programmer.

As PL9 and MACE are only available under FLEX 9 we have shipped them on 5" SS/SD disks....PL9 is 40 TRACK, as we could not fit all of the library files onto a 35 track disk.

I have also enclosed overview of MACE/XMACE and PL9/MATHS PACKAGE. We will be offering MACE and XMACE in a combined package for $98.00 and PL9 with its associated Library and MATHS PACKAGE for $198.00. These prices include air mail postage to the U.S.

We have supplied SSB/FLEX/DS-9 versions of the ALL-IN-TWO software. To our knowledge we are the only manufacturer offering an EPROM programmer for OS-9 systems...a mention of this would be appreciated.

Perhaps a bit of history on PL/9 is in order. One of our company's main lines of work is industrial and process control which as you know is very heavy on the I/O side of things. Back in 1976 we got stuck into the 6800 at the assembly language level as there were no really good compilers about in those days and we did not want to get saddled with putting run-time packages in every system we sold. Then SPL/M (by T. Crowley/Programma Int'l) came out in 1979 and ... W O W ! ... did this cut down the time it took us to produce structured control arguments. We migrated to the 6809 back in 1980 and there was absolutely nothing similar to SPL/M for the 6809 at that time. 'C' and 'NATIVE' Pascals were unheard of in those days. So we had one of our consultants write PL/9 for us which was a cross between SPL/M, Pascal and Assembler. We have found this 'language' to be one of the most code efficient compilers we have used to date. Thus far it has outperformed every version of 'C' we have bought, and every version of Pascal as well, in terms of compactness of code and execution speed. You will have to try it to believe it!

A few points must be kept in mind when 'criticising' the fact that PL/9 is not a standard language when compared with 'C' and Pascal is that it is written for people who normally are forced to use assembler to get the speeds they need in control I/O but would like to have a higher level language for the control arguments that A. does not require a run time package, B. produces ROMable code and C. produces position independent code.

We look forward to receiving your critique in due course.

Yours faithfully,

William C. Dickinson
DIRECTOR

PL/9

A Programming Language for the 6809

PL/9 is a new language designed to be a step up from assembly language, retaining most of the flexibility and speed of the latter but making programs much shorter and more readable. It is a structured language loosely based upon Pascal and PL/M and is particularly suitable for control applications. The program contains an editor, the compiler and a trace/debug package, all of which reside in memory together, thus making for very fast program generation and testing. The compiler produces pure machine code requiring no run-time package and will compile into memory or onto disc. Functions not supported directly by the compiler, such as disk drivers or floating-point arithmetic, can easily be included in PL/9 programs; a number of such functions are available separately. All programs produced by PL/9 are position-independent and ROM-able, making the language highly suitable for applications such as numerical control or intelligent instruments.

Summary of features

1. Built-in comprehensive editor; identical in operation to that of MACE and generally similar to the TSC editor but easier to use.
2. Built-in trace/debug package allowing programs to be run under the supervision of PL/9. Control can thus be regained if things go wrong, variables can be examined and programs can be "single stepped" a source line at a time.
3. Structured high-level language similar to PL/M, Pascal or C but oriented towards problems that require the speed and flexibility of assembler but the ease of use of a high-level language.
4. Compiler generates compact, position-independent, ROM-able code that requires no run-time support.
5. Clear textual error messages. Compilation stops at an error, giving the programmer the choice of continuing or returning to the editor.
6. Direct compilation into memory, with an optional offset, or into an object file on disc.
7. Date printout on each listing page, as an aid to documentation.
8. Library modules can be incorporated to perform defined tasks, thus preventing source programs from getting too large. A comprehensive subroutine library is supplied with PL/9.
9. The MACE assembler can generate PL/9 library modules directly, allowing critical parts of a program to be coded in assembler.

MACE

M6809 Assembler with Co-resident Editor

MACE is a complete editor/assembler system for the 6809. The editor is very quick and easy to use; it loads, saves finds, changes, appends comments, prints lines neatly formatted, passes commands to FLEX and calls the assembler. The assembler has 8-character labels, local labels, cross-reference, multiple file assembly, listing to terminal, printer or disc file, partial assembly listing, symbol table only, object to disc or memory, extra convenience mnemonics, date printout, intelligent error messages, and resides in memory with the editor, source file and symbol table. There is no easier way of writing and testing 6809 assembler programs!

Summary of features

1. Built-in comprehensive editor; generally similar to the TSC editor but easier to use. The editor "pretty-prints" assembly source, making it unnecessary to put in extra spaces or tabs.
2. 8-character global labels to aid program readability.
3. Re-usable numeric local labels to save having to invent symbol names. These labels do not appear in the symbol table.
4. Optional cross-referenced symbol table.

5. All assembly options are defined in the command line, obviating the need to edit the file for different options.

6. Clear textual error messages. Assembly stops at an error, giving the programmer the choice of continuing or returning to the editor.

7. Multiple file assembly mode, dispensing with linking loader.

8. Direct assembly to memory, with an optional offset, or into an object file on disc.

9. Date printout on each listing page, as an aid to documentation.

10. The assembler accepts all 6800/6801 mnemonics as well as several extra convenience mnemonics such as INY, CLRD, etc.

10. The assembler can generate modules for use by the PL/9 compiler as efficient library routines.

SCIENTIFIC FUNCTIONS PACKAGE

This package provides six common scientific functions, viz. natural logarithm, exponential, sine, cosine, tangent and arctangent, and is supplied as a set of PL/9 procedures. The precision of the functions varies but is generally better than 5 significant decimal digits. The package also includes the complete floating-point arithmetic package, which need not therefore be purchased separately. A binary file is included that can be used with assembly-language programs. All routines are position-independent and re-entrant, and the total size of the combined arithmetic and scientific packages is less than 4K bytes.

Summary of features

1. Routines take about 25,000 machine cycles to execute (except for tangent, 50,000 cycles) and are position-independent and re-entrant. Parameters are passed as pointers on the stack.

2. PL/9 source supplied, requiring floating-point arithmetic pack. System-independent object file also supplied.

3. Error trapping indicates when illegal argument is passed. NO values are modified in this case.

FLOATING-POINT ARITHMETIC PACKAGE

This package contains floating-point add, subtract, multiply, divide and square root routines, together with utilities that convert from integer to floating-point, floating-point to integer, ASCII to floating-point and floating-point to ASCII. A number of other useful routines are also provided. The package is designed to be used as a PL/9 library module but can equally well be used in assembly-language programs. The routines operate on numbers whose addresses are passed on the stack; no RAM other than the stack is required. The entire package occupies less than 2k bytes and is highly modular, allowing unwanted sections to be removed in order to save space. Note: This package is supplied as a PL/9 library module and in source form compatible only with the MACE assembler.

Summary of features

1. Floating-point values occupy 4 bytes each, giving a range of ±1E-38 to ±1E37 and about 7 significant (decimal) digits.

2. Routines are fast.

ADD	800	machine cycles
SUBTRACT	800	" "
MULTIPLY	1000	" "
DIVIDE	3000	" "
SQUARE ROOT	6000	" "

3. Conversion routines allow numbers to be input and output in a wide range of formats, e.g. 1234, .052, 3.6E12, -8.4316E-4 etc.

4. Routines are position-independent and re-entrant. Parameters are passed as addresses on the stack.

5. Assembler source supplied (requires the MACE assembler).

6. Compatibility with PL/9 or assembly language.

XMACE

6800/01 Cross-Assembler with Co-resident Editor

XMACE is a complete editor/assembler system, running on the 6809 and generating code for the 6800 or the 6801. It is virtually identical in use to the 6809 version, MACE. The editor is very quick and easy to use; it loads, saves finds, changes, appends comments, prints lines neatly formatted, passes commands to FLEX and calls the assembler. The assembler has 8-character labels, local labels, cross-reference, multiple file assembly, listing to terminal, printer or disc file, partial assembly listing, symbol table only, object to disc or memory, extra convenience mnemonics, date printout, intelligent error messages, and resides in memory with the editor, source file and symbol table.

Summary of features

1. Built-in comprehensive editor; generally similar to the TSC editor but easier to use. The editor "pretty-prints" assembly source, making it unnecessary to put in extra spaces or tabs.

2. 8-character global labels to aid program readability.

3. Re-usable numeric local labels to save having to invent symbol names. These labels do not appear in the symbol table.

4. Optional cross-referenced symbol table.

5. All assembly options are defined in the command line, obviating the need to edit the file for different options.

6. Clear textual error messages. Assembly stops at an error, giving the programmer the choice of continuing or returning to the editor.

7. Multiple file assembly mode, dispensing with linking loader.

8. Direct assembly to memory, with an optional offset, or into an object file on disc.

9. Date printout on each listing page, as an aid to documentation.

10. A switch allows the programmer to select "6800-only" or "6800 and 6801" mnemonics.

As a technician in the video field I have always been interested in computers and there graphic capabilities. After four years of psuedo graphics someone has finaly come out with a graphics board for the SWTPC's 6800. COLORAMA 50 tm by PERCOM DATA. Percom Data is a company that I could recommend to all of you who are looking for an S-50 buss graphics board. The board is of high quality and are assembled so that all you need do is configure the various jumpers and plug it in. The manual comes complete with how to information a schematic and parts location and is

printed on heavy bond so it should last a good long time for the new comer. The board is very easy to program, from basic you can simply POKE what ever you want to the on board ram. I have included a very simple program that I call ECHO. The programs main purpose is to produce titling for a video recorder (has any one priced a character generator lately?). Now by the time you get to read this I will have expanded this program considerably. If anyone is interested I will make it available if they send an SASE or a 5 inch disk and return postage. To use this program all you need is a 6800 S-50 computer the COLORAMA 50 a CRT and a TV or monitor. One last comment, I would like to thank all the companies that I have had the honor of dealing with that make computing so trouble free. SWTPC, DIGITAL RESEARCH COMPUTERS of Texas, COMPUWARE CORP., WILLIAMS ELECTRONICS of Edison NJ, and TSC for there excellent software, oh yes, Thank you 68 MICRO JOURNAL for giving me this opertunity.

Dick Melbarde

P.S. Phone # 201-852-9389

105 CHRISTOPHER ST.
HACKETTSTOWN NJ 07840

```
100 REM ************************************************
200 REM * THIS PROGRAM WILL ECHO A CHARACTER FROM       *
300 REM * THE CONTROL TERMINAL TO THE COLORAMA VIDEO    *
400 REM * BOARD.  WRITTEN BY MR. D. MELBARDE            *
500 REM *                105 CHRISTOPHER ST             *
600 REM *                HACKETTSTOWN NJ 07840          *
700 REM * RIGHT TO PUBLISH GRANTED                      *
800 REM ************************************************
900 REM * COLORAMA TM OF PERCON DATA                    *
1000 REM ***********************************************
1100 REM * VIDEO RAM ADDRESSED AT C000H                 *
1110 REM ***********************************************
1200 CL$=CHR$(27)+CHR$(69)
1300 C=49152
1400 FOR Z=1 TO 512
1500 POKE C,32
1600 C=C+1
1700 NEXT Z

1800 PRINT CL$
1900 REM ***********************************************
2000 Y1=49152 : Y=Y1
2100 R=0 : CU=0 : POKE Y,159
2200 A$=INCH$(0)
2300 A=ASC(A$)
2400 REM ROW COUNTER = R
2500 IF R<0 OR R>31 THEN R=0
2600 IF A=8 THEN 3900 : REM BACK SPACE
2700 IF A=10 THEN 4600 : REM LINE FEED
2800 IF A=13 THEN 5200 : REM CARRIAGE RETURN
2900 IF A=32 THEN 3200 : REM PRINT SPACE
3000 IF A=127 THEN 5900 : REM TURN ON/OFF CURSOR
3100 IF A>=65 THEN A=A-64 : GOTO 3200 : REM START ALPHA FROM 1 > 27
3200 POKE Y,A
3300 PRINT R;Y;A
3400 Y=Y+1
3500 R=R+1
3600 POKE Y,159
3700 GOTO 2200
3800 REM ***********************************************
3900 REM BACK SPACE
4000 POKE Y,32

4100 Y=Y-1
4200 R=R-1
4300 POKE Y,159
4400 GOTO 2200
4500 REM ***********************************************
4600 REM LINE FEED
4700 POKE Y,32
4800 Y=Y+32
4900 POKE Y,159
5000 GOTO 2200
5100 REM ***********************************************
5200 REM CARRIAGE RETURN
5300 POKE Y,32
```

ECHO VERSION 4 XBASIC   3-15-82   PAGE 2

```
5400 Y=Y-R
5500 R=0
5600 POKE Y,159
5700 GOTO 2200

5800 REM ***********************************************
5900 REM DELETE = DELETE OR TURN OFF/ON CURSOR
6000 IF CU=0 THEN 6200
6100 IF CU=1 THEN 6300
6200 POKE Y,32 : CU=1 : REM CURSOR OFF
6300 POKE Y,159 : CU=0 : REM CURSOR ON
6400 GOTO 2200
```

[redacted]

```
                         nam    INDEX
                         ttl    program to generate file index for diskettes
A000                     org    $a000
                    *
                    * INDEX program to generate file index for diskettes
                    *
                    * copyright 1981 by Tom Weaver
                    *                       925 N. Emery
                    *                       Norman, Okla 73069
                    *                       (405) 360-6856
                    *
          C080      linbuf   equ   $c080     flex9 line buffer
          CC14      bufpnt   equ   $cc14     flex9 line buffer pointer
          CD03      warms    equ   $cd03     flex9 warm start address
          CD0C      inch     equ   $cd0c     flex9 terminal character input
          CD1E      pdata    equ   $cd1e     flex9 terminal string output
          CD3F      rpterr   equ   $cd3f     flex9 error report
          CD4B      flex9s   equ   $cd4b     flex9 as a subroutine entry point
          D406      fms      equ   $d406     file management system entry point
          0001      opnin    equ   1
          0002      opnout   equ   2
          0004      close    equ   4
          0006      opndir   equ   6
          0007      getdir   equ   7
          000C      delete   equ   12
          0010      opninf   equ   16
                    *
                    *
                    *
A000 8E   A53B      index    ldx   #remark
A003 BD   CD1E               jsr   pdata     give prompt for drive 1
A006 BD   CD0C               jsr   inch      wait until ready
A009 8E   A6BC               ldx   #tmpfil
A00C BD   D406               jsr   fms       delete any garbage
A00F 8E   A6AD               ldx   #filout
A012 BD   D406               jsr   fms       open initial work file
A015 1026 2DC6               lbne  rpterr
                    *
A019 8E   A3BF      mainlp   ldx   #prompt
A01C BD   CD1E               jsr   pdata     output prompt for new disk
A01F BD   CD0C               jsr   inch      input reply
A022 81   53                 cmpa  #'S       stop ?
A024 27   68                 beq   done      no
A026 81   73                 cmpa  #'s       stop ?
A028 27   64                 beq   done      yes
A02A 8E   A56D               ldx   #filin
A02D 86   10                 lda   #opninf
A02F A7   84                 sta   ,x
A031 BD   D406               jsr   fms       open sys info rec
A034 1026 0161               lbne  error     disk read error - attempt recovery
A038 86   07                 lda   #getdir
A03A A7   84                 sta   ,x
A03C BD   D406               jsr   fms       read sys info record
A03F 1026 015A               lbne  error     disk read error - attempt recovery
A043 EC   1F                 ldd   31,x      disk number
A045 8E   A5A5               ldx   #dsknmo
A048 BD   A23C               jsr   cvtnum    convert binary to ASCII
A04B 8E   A56D               ldx   #filin
A04E 86   06                 lda   #opndir
A050 A7   84                 sta   ,x
A052 BD   D406               jsr   fms       open directory
A055 1026 0144               lbne  error     Disk read error - attempt recovery
                    *
A059 86   07        inloop   lda   #getdir
A05B A7   84                 sta   ,x
A05D BD   D406               jsr   fms       read directory entry
A060 27   09                 beq   inok
A062 A6   01                 lda   1,x       load error code
A064 81   08                 cmpa  #8        end of file ?
A066 27   B1                 beq   mainlp    yes
A068 7E   A1F9               jmp   error     no - attempt recovery
A06B 6D   04        inok     tst   4,x       directory entry used ?
A06D 27   AA                 beq   mainlp    no - never
A06F 2B   E8                 bmi   inloop    ye - deleted
A071 CE   A2A0               ldu   #size
A074 8E   A582               ldx   #filin+21 size of file
A077 BD   A23C               jsr   cvtnum    convert binary to ASCII
A07A CE   A571               ldu   #filin+4
A07D 8E   A6AD               ldx   #filout
A080 BD   A1A1               jsr   outit     output file name
A083 CE   A2A0               ldu   #size     jsr outit
A086 BD   A1A1               jsr   outit     output file size and disk number
A089 8E   A56D               ldx   #filin
A08C 20   CB                 bra   inloop    go read next entry for this disk
                    *
                    *
A08E 8E   A6AD      done     ldx   #filout
A091 86   04                 lda   #close
A093 A7   84                 sta   ,x
A095 BD   D406               jsr   fms       close output file
A098 8E   A35B               ldx   #master
A09B BD   CD1E               jsr   pdata     get pgm disk back in drive 0
A09E BD   CD0C               jsr   inch
A0A1 CE   C080               ldu   #linbuf   flex9 line buffer
A0A4 8E   A2DA               ldx   #srtcal
A0A7 BF   CC14               stx   bufpnt    flex9 line buffer pointer
A0AA EC   81        callop   ldd   ,x++
A0AC ED   C1                 std   ,u++
A0AE C1   0D                 cmpb  #$0d      carriage return ?
A0B0 26   F8                 bne   callop
A0B2 BD   CD4B               jsr   flex9s    call flex sort as a subroutine
A0B5 5D                      tstb            everything ok ?
A0B6 1026 0179               lbne  badsrt    no
A0BA 86   0C                 lda   #delete
A0BC 8E   A6AD               ldx   #filout
A0BF A7   84                 sta   ,x
A0C1 BD   D406               jsr   fms       delete initial work file
A0C4 8E   A3F8      asktit   ldx   #title    prompt for title/date
A0C7 BD   CD1E               jsr   pdata
A0CA 8E   A4E0               ldx   #headr2
A0CD C6   50                 ldb   #80       maximum title length
```

```
A0CF BD   CD0C   titlop  jsr    inch        input char from terminal
A0D2 81   08             cmpa   #$08        backspace ?
A0D4 1027 012A           lbeq   backup      yes
A0D8 81   0D             cmpa   #$0D        return
A0DA 27   09             beq    titdun      yes - end of title
A0DC 81   1D             cmpa   #$1d        line cancel ?
A0DE 27   E4             beq    asktit      yes - start title over
A0E0 A7   80             sta    ,x+         save character
A0E2 5A                  decb               title too long ?
A0E3 26   EA             bne    titlop      no - get more
A0E5 CC   0000   titdun  ldd    #$0000
A0E8 ED   81             std    ,x++
A0EA 86   04             lda    #4
A0EC A7   84             sta    ,x
A0EE 8E   A570           ldx    #innam-1
A0F1 CE   A2AB           ldu    #sorted
A0F4 BD   A294           jsr    movnam      name of sorted work file
A0F7 8E   A6B0           ldx    #outnam-1
A0FA CE   A2B7           ldu    #dskidx
A0FD BD   A294           jsr    movnam      name of final file
A100 8E   A6AD           ldx    #filout
A103 86   0C             lda    #delete
A105 A7   84             sta    ,x
A107 BD   D406           jsr    fms         delete any old INDEX
A10A 8E   A6B0           ldx    #outnam-1
A10D CE   A2B7           ldu    #dskidx
A110 BD   A294           jsr    movnam      name of final file
A113 8E   A6AD           ldx    #filout
A116 86   02             lda    #opnout
A118 A7   84             sta    ,x
A11A BD   D406           jsr    fms         open output file
A11D BD   A211           jsr    phdr        output top of page header
A120 86   01             lda    #opnin
A122 A7   84             sta    ,x
A124 BD   D406           jsr    fms         open sorted work file
A127 C6   08             ldb    #8
A129 CE   A422           ldu    #line+10
A12C BD   A208   finnam  jsr    indata      read file name
A12F CE   A428           ldu    #line+19
A132 C6   0D             ldb    #13
A134 BD   A208   finish  jsr    indata      read file name ext, size, and disk
A137 1F   32             tfr    u,y         save end of line pointer
A139 BD   D406           jsr    fms         read carriage return
A13C CE   A 32           ldu    #line+10    point to start of file name
A13F BD   D406   dupfin  jsr    fms         read char of new file name
A142 26   4C             bne    eofend      must be end of file
A144 A7   C8 E0          sta    -22,u       save this character
A147 A1   C0             cmpa   ,u+         does it match prev file name char ?
A149 26   1B             bne    notsam      no
A14B 5C          dupinc  incb
A14C C1   08             cmpb   #8
A14E 27   12             beq    ovrdot      skip over period
A150 C1   0B             cmpb   #11
A152 26   EB             bne    dupfin      check rest of name
A154 C6   05             ldb    #5
A156 BD   D406   duppit  jsr    fms         ignore file size
A159 5A                  decb
A15A 26   FA             bne    duppit
A15C 1F   23             tfr    y,u         end of line pointer
A15E C6   05             ldb    #5          just append disk number
A160 20   D2             bra    finish
                 *
A162 33   41     ovrdot  leau   1,u         skip over period
A164 20   D9             bra    dupfin
                 *
A166 88   20     notsam  eora   #$20        check upper/lower case
A168 A1   C7             cmpa   -9,u        match now ?
A16A 26   04             bne    newnam      no
A16C 33   41             leau   1,u         yes - continue comparison
A16E 20   DB             bra    dupinc
                 *
A170 34   44     newnam  pshs   b,u         save current position
A172 CC   0004           ldd    #$0004
A175 ED   A4             std    ,y
A177 CE   A418           ldu    #line
A17A BD   A221           jsr    outlin      output file info
A17D 7A   A29E           dec    curlin      number lines left on this page
A180 26   03             bne    newlin
A182 BD   A216           jsr    endpag      skip lines and print header
A185 CE   A422   newlin  ldu    #line+10
A188 C6   0C             ldb    #12
A18A A6   C8 EA  newfil  lda    -22,u       copy name of new file
A18D A7   C0             sta    ,u+
A18F 5A                  decb
A190 26   F8             bne    newfil
A192 35   44             puls   b,u         restore pointers
A194 33   41             leau   1,u
A196 50                  negb
A197 CB   07             addb   #7          length of name to finish
A199 27   96             beq    finish-5
A19B 2A   97             bpl    finnam
A19D CB   0D             addb   #13
A19F 20   93             bra    finish
                 *
A1A1 C6   0B     outsi   ldb    #11         process 11 characters
A1A3 A6   C0     outchr  lda    ,u+         pick up character
A1A5 26   02             bne    nonull
A1A7 86   20             lda    #$20        replace null by blank
A1A9 BD   D406   nonull  jsr    fms         write to disk file
A1AC 5A                  decb               number of char to go
A1AD 26   F4             bne    outchr
A1AF 39                  rts
                 *
A1B0 CC   0D04   eofend  ldd    #$0D04
A1B3 ED   A4             std    ,y
A1B5 CE   A418           ldu    #line
A1B8 BD   A221           jsr    outlin      out ut last line of file info
A1BB F6   A29E           ldb    curlin
A1BE CB   0B             addb   #11
A1C0 8E   A6AD           ldx    #filout
A1C3 86   0D     endskp  lda    #$0d
A1C5 BD   D406           jsr    fms         space to end of page
A1C8 5A                  decb
A1C9 26   F8             bne    endskp
A1CB 86   04             lda    #close
A1CD A7   84             sta    ,x
A1CF B7   A56D           sta    filin
A1D2 BD   D406           jsr    fms         close final output file
A1D5 8E   A56D           ldx    #filin
A1D8 BD   D406           jsr    fms         close sorted work file
A1DB 86   0C             lda    #delete
A1DD A7   84             sta    ,x
A1DF BD   D406           jsr    fms         delete sorted work file
A1E2 CE   C080           ldu    #linbuf     flex line buffer
A1E5 8E   A2C3           ldx    #printt
A1E8 BF   CC14           stx    bufpnt      flex line buffer pointer
A1EB EC   81     endlop  ldd    ,x++
A1ED ED   C1             std    ,u++
A1EF 81   0D             cmpa   #$0d        return ?
A1F1 26   F8             bne    endlop
A1F3 BD   CD4B           jsr    flex9x      call flex to print index
A1F6 7E   CD03           jmp    warms       finished
                 *
A1F9 8E   A392   error   ldx    #errmsg
A1FC BD   CD1E           jsr    pdata       disk read error message
A1FF 7E   A019           jmp    mainlp      try it again
                 *
A202 30   1F     backup  leax   -1,x        backspace one position
A204 5C                  incb
A205 7E   A0CF           jmp    titlop
                 *
A208 BD   D406   indata  jsr    fms         read a char from disk
A20B A7   C0             sta    ,u+
A20D 5A                  decb
A20E 26   F8             bne    indata      go read more
A210 39                  rts
                 *
A211 CE   A4C7   phdr    ldu    #header     top of page header
A214 20   03             bra    *+5
A216 CE   A45C   endpag  ldu    #skip       don't print on perforation
A219 8D   06             bsr    outlin
A21B 86   35             lda    #53         number of lines left per page
A21D B7   A29E           sta    curlin
A220 39                  rts
                 *
A221 8E   A6AD   outlin  ldx    #filout     output FCB address
A224 A6   C0             lda    ,u+
A226 81   04             cmpa   #4          end of line ?
A228 27   05             beq    outdun      yes
A22A BD   D406           jsr    fms         output a character
A22D 20   F5             bra    outlin+3
A22F 8E   A56D   outdun  ldx    #filin      input FCB address
A232 39                  rts
                 *
A233 8E   A30E   badsrt  ldx    #srtbad
A236 BD   CD1E           jsr    pdata       sort error message
A239 7E   CD03           jmp    warms       give up
                 *
A23C 4F          cvtnum  clra               turn off significance
A23D B7   A29F           sta    filler
A240 C6   30             ldb    #'0
A242 8C   2710   cvtl1   cmpx   #10000      over ten thousand ?
A245 2D   0C             blt    cvttho      no
A247 F7   A29F           stb    filler      turn on significance
A24A 8A   30             ora    #'0         force to print
A24C 30   89 D8F0        leax   -10000,x    subtract ten thousand
A250 4C                  inca
A251 20   EF             bra    cvtl1
A253 A7   C4     cvttho  sta    ,u          store ten thousands digit
A255 B6   A29F           lda    filler
A258 8C   03E8   cvtl2   cmpx   #1000       over one thousand
A25B 2D   0C             blt    cvthun      no
A25D F7   A29F           stb    filler      turn on significance
A260 8A   30             ora    #'0         force to print
A262 30   89 FC18        leax   -1000,x     subtract a thousand
A266 4C                  inca
A267 20   EF             bra    cvtl2
A269 A7   41     cvthun  sta    1,u         store thousands digit
A26B B6   A29F           lda    filler
A26E 8C   0064   cvtl3   cmpx   #100        over a hundred ?
A271 2D   0B             blt    cvtten      no
A273 F7   A29F           stb    filler      turn on significance
A276 8A   30             ora    #'0         force to print
A278 30   88 9C          leax   -100,x      subtract a hundred
A27B 4C                  inca
A27C 20   F0             bra    cvtl3
A27E A7   42     cvtten  sta    2,u         store hundreds digit
A280 B6   A29F           lda    filler
A283 8C   000A   cvtl4   cmpx   #10         over ten ?
A286 2D   07             blt    cvtone      no
A288 8A   30             ora    #'0         force to print
A28A 30   16             leax   -10,x       subtract ten
A28C 4C                  inca
A28D 20   F4             bra    cvtl4
A28F 30   8B     cvtone  leax   d,x         combine digits and force units to pr
A291 AF   43             stx    3,u         store tens and units digits
A293 39                  rts
                 *
A294 C6   0C     movnam  ldb    #12         move 12 characters
A296 A6   C0             lda    ,u+
A298 A7   80             sta    ,x+
A29A 5A                  decb
A29B 26   F9             bne    movnam+2    go move more
A29D 39                  rts

                 *
                 * FCB directory offsets
                 *
                 *   0   -function code
                 *   3   -drive number
                 *  4-11 -file name
                 * 12-14 -file extension
                 * 21-22 -file size
                 *
                 * - - - - - - - - - - - - - - - - - - - -
                 *
                 * work file format
                 *
                 *  1-8  -file name
                 *  9-11 -file extension
                 * 12-16 -file size
                 * 17-21 -disk number
                 *  22   -carriage return
                 *
                 * - - - - - - - - - - - - - - - - - - - -
                 *
                 * final output file format
                 *
                 * 11-18 -file name
                 *  19   -period
                 * 20-22 -file extension
                 * 23-27 -file size
                 * 28-32 -first disk number
                 * 33-37 -second disk number
                 *  ...  -additional disk numbers
                 *   ?   -carriage return
                 *
                         pag
A29E 00          curlin  fcb    0
A29F 00          filler  fcb    0
A2A0             size    rmb    5
A2A5             diskno  rmb    5
A2AA 0D                  fcb    $0d
                 *
A2AB 01          sorted  fcb    1
A2AC 51 20 50 5A         fcc    '[illegible]'
A2B0 56 4A 47 57
A2B4 51 48 42
A2B7 01          dskidx  fcb    1
A2B8 44 49 53 4B         fcc    'DISKINDXLST'
A2BC 49 4E 44 58
A2C0 4C 53 54
                 *
A2C3 30 2E 53 20 printt  fcc    '0.S 0.LIST 1.DISKINDX'
A2C7 30 2E 4C 49
A2CB 53 54 20 31
A2CF 2E 44 49 53
A2D3 4B 49 4E 44
A2D7 58
A2D8 0D 0D               fcb    $0d,$0d
A2DA 30 2E 50 53 srtral  fcc    '0.PSORT 0.SORTSPEC [illegible]'
A2DE 4F 52 54 20
A2E2 30 2E 53 4F
A2E6 52 54 53 50
A2EA 45 43 20 2B
A2EE 31 2E 51 2D
A2F2 58 5A 56 4A
A2F6 47 57 2E 51
A2FA 48 42 29 20
A2FE 31 2E 5A 2D
A302 58 2D 2D 58
A306 5A 51 2E 56
A30A 2D 4A
A30C 0D 0D               fcb    $0d,$0d
                 *
A30E 53 6F 72 74 srtbad  fcc    'Sort bombed - index aborted.'
A312 20 62 6F 6D
A316 62 65 64 20
A31A 2D 20 69 6E
A31E 64 65 78 20
A322 61 62 6F 72
A326 74 65 64 2E
A32A 04                  fcb    4
A32B 49 6E 73 65 remark  fcc    'Insert clean disk in drive 1 and press return.
A32F 72 74 20 63
A333 6C 65 61 6E
A337 20 64 69 73
A33B 6B 20 69 6E
A33F 20 64 72 69
A343 76 65 20 31
A347 20 61 6E 64
A34B 20 70 72 65
```

```
A34F 73 73 20 72
A353 65 74 75 72
A357 6E 2E 20
A35A 04                     fcb   4
A35B 49 6E 73 65  master    fcc   'Insert program disk back in drive 0 '
A35F 72 74 20 70
A363 72 6F 67 72
A367 61 6D 20 64
A36B 69 73 6B 20
A36F 62 61 63 6B
A373 20 69 6E 20
A377 64 72 69 76
A37B 65 20 30 20
A37F 61 6E 64 20            fcc   'and press return. '
A383 70 72 65 73
A387 73 20 72 65
A38B 74 75 72 6E
A38F 2E 20
A391 04                     fcb   4
A392 44 69 73 6B  errmsg    fcc   'Disk read error.  Reseat disk and try again.'
A396 20 72 65 61
A39A 64 20 65 72
A39E 72 6F 72 2E
A3A2 20 20 52 65
A3A6 73 65 61 74
A3AA 20 64 69 73
A3AE 6B 20 61 6E
A3B2 64 20 74 72
A3B6 79 20 61 67
A3BA 61 69 6E 2E
A3BE 04                     fcb   4
A3BF 49 6E 73 65  prompt    fcc   'Insert disk in drive 0 and press return.'
A3C3 72 74 20 64
A3C7 69 73 6B 20
A3CB 69 6E 20 64
A3CF 72 69 76 65
A3D3 20 30 20 61
A3D7 6E 64 20 70
A3DB 72 65 73 73
A3DF 20 72 65 74
A3E3 75 72 6E 2E
A3E7 20 20
A3E9 45 6E 74 65            fcc   'Enter "S" to stop:
A3ED 72 20 22 53
A3F1 22 20 74 6F
A3F5 20 73 74 6F
A3F9 70 3A 20 20
A3FD 04                     fcb   4
A3FE 44 61 74 65  title     fcc   'Date/title:
A402 2F 54 69 74
A406 6C 65 3A 20
A40A 20
A40B 0                      fcb   4
                  *
A40C 20 20 20 20            fcc   '
A410 20 20 20 20
A414 2E 20 20 20
A418 20 20 20 20  line      fcc   '
A41C 20 20 20 20
A420 20 20 20 20
A424 20 20 20 20
A428 20 20 2E 20
A42C 20 20 20 20
A430 20 20 20 20
A434 20 20 20 20
A438 20 20
A43A                        rmb   130
                  *
A4BC 0D 0D 0D 0D  skip      fcb   $0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d
A4C0 0D 0D 0D 0D
A4C4 0D 0D 0D
A4C7 20 20 20 20  header    fcc   '          File name   Size   Disks
A4CB 20 20 20 20
A4CF 20 20 20 46
A4D3 69 6C 65 20
A4D7 6E 61 6D 65
A4DB 20 20 20 53
A4DF 69 7A 65 20
A4E3 20 20 44 69
A4E7 73 6B 73 20
A4EB 20 20
A4ED 20 20 20 20  headr2    fcc   '
A4F1 20 20 20 20
A4F5 20 20 20 20
A4F9 20 20 20 20
A4FD 20 20 20 20
A501 20 20 20 20
A505 20 20 20 20
A509 20 20 20 20
A50D 20 20 20 20
A511 20 20 20 20
A515 20 20 20 20            fcc   '
A519 20 20 20 20
A51D 20 20 20 20
A521 20 20 20 20
A525 20 20 20 20
A529 20 20 20 20
A52D 20 20 20 20
A531 20 20 20 20
A535 20 20 20 20
A539 20 20 20 20
A53D 20 20 20 20
A541 20 20 20 20            fcc   '
A545 20 20 20 20
A549 20 20 20 20
A54D 20 20 20 20
A551 20 20 20 20
A555 20 20 20 20
A559 20 20 20 20
A55D 20 20 20 20
A561 20 20 20 20
A565 20 20 20 20
A569 20 20 20 20
                  *
A56D [illegible]  filin     fcb   [illegible]
A571              innam     rmb   [illegible]
                  *
A6AD [illegible]  filout    fcb   [illegible]
A6B1 5A 20 50 20  outnam    fcc   [illegible]
A6B5 20 50 5A 51
A6B9 56 20 4A
A6BC [illegible]  tmpfil    fcb   [illegible]
A6C0 5A 20 50 20            fcc   [illegible]
A6C4 20 50 5A 51
A6C8 56 20 4A
A6CB                        rmb   305
                  *
                            end   index
```

SYMBOL TABLE:

```
[illegible]  [illegible]   alend     A1[illegible]   backup    A2[illegible]   batert    A233   bufent    CC[illegible]
[illegible]  [illegible]   [illegible] [illegible]   [illegible] A2[illegible] cvthun  A2[illegible] cvtll  A2[illegible]
[illegible]  A258          cvtl2     A2AE            cvtl4     A2[illegible]   cvtinum   A2[illegible]   cvtone    A2[illegible]
[illegible]  A2[illegible] cvttho    A253            delete    [illegible]     diskno    A2A5   done      A0BE
[illegible]  A2[illegible] dupfin    A1[illegible]   dupinc    A14B            duppit    A156   endlop    A1[illegible]
[illegible]  A2[illegible] [illegible] A1C3          errm g    A2[illegible]   error     A1F9   filin     A56D
[illegible]  [illegible]   filout    A6AD            finish    A1[illegible]   finnam    A12C   [illegible] CD4B
[illegible]  D406          getdir    [illegible]     header    A4C7            headr2    A4ED   inch      CD0C
[illegible]  [illegible]   index     [illegible]     [illegible] [illegible]   innam     A571   [illegible] [illegible]
[illegible]  [illegible]   line      A[illegible]    [illegible] [illegible]   master    A35B   [illegible] [illegible]
[illegible]  [illegible]   newlin    A105            [illegible] A170          nonull    A1A9   notend    A1[illegible]
[illegible]  [illegible]   [illegible] [illegible]   opnin     0010            opnout    0002   outchr    A1A3
[illegible]  A2[illegible] outit     A1A1            outlin    A2[illegible]   outnam    A6B1   ovrdot    A162
[illegible]  CD1E          phdr      A211            printt    A2C3            prompt    A3BF   remark    [illegible]
[illegible]  [illegible]   size      A2A0            skip      A4BC            sorted    A2AB   [illegible] A30[illegible]
[illegible]  A2DA          [illegible] [illegible]   title     A3FE            titlop    A[illegible]   tmpfil    A6BC
[illegible]  [illegible]
```

technical systems consultants, inc.

NEW PRODUCT BULLETIN
Release Date: IMMEDIATELY
Contact: Dan Stankewicz
Phone: (919)493-1451

FORTRAN Compiler for 6809

For the first time, a 6809 ANSI FORTRAN-77 compiler is available. Running under FLEX™ and UniFLEX™, the native code compiler produces assembler language output compatible with the Technical Systems Consultants relocating assembler and linkage editor.

Technical Systems Consultants' Fortran conforms to ANSI FORTRAN-77 (ANSI X3.9-1978) subset of the FORTRAN language, with the following exceptions:

The INTRINSIC and SAVE statements are ignored.
The EQUIVALENCE statement is not implemented.
The BACKSPACE statement is only allowed on direct access files.
The ENDFILE statement performs no useful function.
Statement functions are not supported.
Variable names only two of any length with 7 characters significant.
All keywords are reserved names.
Direct access files are not available under FLEX.

In addition, Technical Systems Consultants' Fortran contains some features of the full FORTRAN language, most notably bit-directed I/O and expanded form of the OPEN statement. Also included in the extentions are the ability to open any file name, access to command line arguments, and, in the UniFLEX™ version, direct access files.

The FORTRAN library includes modules for 14.8 digit floating point arithmetic, all standard scientific functions, complex file manipulations, runtime trace back feature, and post-mortem dump capability. The compiler also offers many compile time options. These options allow the user to save the assembler source file, prohibit the production of binary, prohibit the invocation of the linkage editor, incorporate post-mortem dump code, and others.

The FORTRAN-77 compiler is available on either 5 or 8 inch diskette for the 6809 FLEX™ Disk Operating System or on 8 inch diskette for the 6809 UniFLEX™ Operating System. A single CPU license for the UniFLEX™ version sells for $450.00 and includes the relocating assembler, linkage editor, and one year of maintenance. The FLEX™ version sells for $375.00 and also includes the relocating assembler and linkage editor. These packages are available off the shelf from Technical Systems Consultants, Inc., 111 Providence Road, Chapel Hill, North Carolina 27514 Telephone (919) 493-1451 or TWX 510-928-0540.

111 Providence Road • Chapel Hill, North Carolina 27514 • (919) 493-1451 • TWX 510-928-0540

83,Castleford Avenue,
London, SE9 2AH
U.K.

21 June 1982

Dear Mr. Williams,

Having bought a new printer, I soon noticed that the TSC Text Processor makes no provision for outputting the necessary control characters to change typeface etc.

The new command, .HX, listed below, allows Hex characters to be sent to the printer:-

```
.HX xx,yy.....
```

This new command replaces .DH,.DW and .DB which are not, in any case, compatible with the printer.
As it stands it is necessary, when using the .CP mode to:-

```
.NC
.HX xx,yy
.CP
```

or, better, to write macros to do the same task. (This is due to the fact that, in the .CP mode, the command line is converted to lower case on being entered in the command buffer. Lower case Hex characters (A...F) are not acceptable to the FLEX GETHEX routine). A fix for this is quite feasible but would require external patch space and did not seem worth the trouble.

The changes are simple enough to be entered by using the MIKBUG (or equivalent) Memory Change facility. The new version can then be saved on FLEX (@200 - 177A, @200 start).
I am still using MINIFLEX. Equivalents for the later version are:-

```
GETHEX:  $AD42
BUFPTR:  $AC14
```

Sincerely

D. Wilkinson

```
1                        NAM    HEX
2                    *TEXT PROCESSOR PRINTER TYPEFACE CHANGE
3                    *EQUATES
4   1211             LONSKP EQU    $1211
5   7894             BUFPTR EQU    $7894
6   713F             GETHEX EQU    $713F
7   B203             OUTCH  EQU    $B203
8   00FA             MEM2   EQU    $00FA

10  89A5                    ORG    $89A5
11  89A5 4B                 FCC    /HX/
    89A6 58
12  89A7 0C 47              FDB    $0C47
13  89A9 48                 FCC    /HX/
    89AA 58
14  89AB 0C 47              FDB    $0C47
15  89AD 48                 FCC    /HX/
    89AE 58
16  89AF 0C 47              FDB    $0C47

18  0C47                    ORG    $0C47
19  0C47 BD 12 11           JSR    LONSKP
20  0C4A FF 78 94           STX    BUFPTR
21  0C4D FE 78 94    HEX1   LDX    BUFPTR
22  0C50 A6 00              LDA A  00,X
23  0C52 81 0D              CMP A  #$0D      CR?
24  0C54 27 0C              BEQ    HEX2
25  0C56 BD 71 3F           JSR    GETHEX    CONVERT TO HEX
26  0C59 DF FA              STX    MEM2
27  0C5B 96 FB              LDA A  MEM2+1    GET HEX CHAR
28  0C5D BD B2 03           JSR    OUTCH     PRINT IT
29  0C60 20 EB              BRA    HEX1
30  0C62 39          HEX2   RTS
31                       END

NO ERROR(S) DETECTED

SYMBOL TABLE:
```

COMPUTERWARE™
Microcomputer Sales and Software

NEWS RELEASE

Computerware™ introduces its MAGIKUBE™ on cassette for the Radio Shack Color Computer.

Magikube™ is a computer version of RUBIK'S CUBE. It honestly is not for anyone who gives up easily! You have the option of scrambling the cube yourself or of several levels of scramble by the Color Computer. Then, the objective is to restore the cube back to its original state. Magikube™ uses the highest level of graphics and, because it is not easy, there is a tape save feature which allows you to save your cube and continue later.

Magikube™ costs only $19.95 and is available directly from Computerware™ at Box 668, 1472 Encinitas Blvd., Encinitas, Ca. 92024, (714)-436-3512.

CLASSIFIED

6800 CPU Card $100.00. AC-30 Cassette interface with documentation $50, DC-3 Controller Card $100.
LEW 1-615-842-4601

•••••

2-MSI 6800 CPU's (46K and 56K),4-8" drives, complete accounting software, SDOS operating system, BASIC, $2,000 for 1 CPU and 2 drives.
Barbara Rush 214-270-0810

•••••

SWTPC M6800,Minifloppy Disk (mini FLEX),CT-82, and more. $3300 or best offer. Call or send S.A.S.E. for details.
Charles Alukonis, 27 Gould St,Paterson, NJ 07503 (201)345-5269

•••••

WANTED SWTPC CT64,DC2,MP09A or MP09B board.
R Steedman, P O Box 98 Bairnsdale,3875 Australia. PH(051)568291

•••••

Hazelwood DM-64A 2Mhz 64K dram board $325. SWTPC MP-8M board $75. PERCOM CIS-30 1200 baud KC-STD cassette interface-$45. Exorcisor bus compatible single board computer 4K RAM, serial port, provision for 4-2716's.
Mark Sicking,(314)878-1730,272-6281

•••••

Hazelwood VC-256 high resolution (256x256) video graphics board for SS-30. Excellent $225.
Sam Green (314)878-4897 evenings

•••••

1980 SWTPC S09 Computer, model 6809, CT82 terminal, SP3 Qume printer,DMF-2 disk drives, software available.
Liz, (512)227-6229

*****

6800,MF-68 2 drvs,CL-4 clock/date,22-MP-S, MP-L, MP-T, MP-8M, GIMIX 32K (40K TOT), clock drivers in ROM, Centronics 301 printer, all TSC software,+Dynamite, and much more. Custom chasis and power supply.
Randy, (702)458-8215 home or (702)798-5616 work

*****

HELP! I have tried running FLEX9 on two computer systems and the same thing happens. The only thing I am switching between systems is the GIMIX 6809 CPU card. The problems are: 1)when I use the TSC text processor, sometimes it doesn't wait for me to enter what are the page limits-it just processes the text without my consent. 2)when I use the TSC editor it cues me twice for a delete of backup file-after I enter 'Y' another 'Y' mysteriously appears-after the pound sign another 'Y' appears 3)when FLEX9 boots up sometimes it prints two DOS cues Do other FLEX9 users have to live with these flukes? I have tried changing the MPU but the same things happen. I have run both systems on FLEX2 without flukey behaviour. Thank you,
Jeff Craig Apt 912-3001 S King Dr, Chicago, Ill 60616 P.S. I have 2 GIMIX mainframes with no.58 floppy disc cards, different RAM cards, different floppy disc drives, and two SOROC IQ 120 terminals.

*****

Please help me locate a cross-assembler that runs on the Apple to produce 6800 code.
Bill Mains, Ms Electron Company, 4213 Phinney North, Seattle,WA 98103

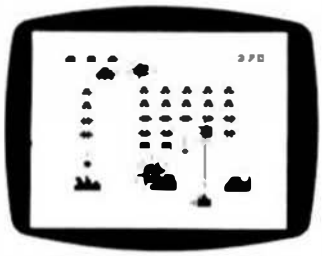
*****

I need communication programs that can run the IBM 3270 and IBM 3780 protocols under UNIFLEX. I am very interested in converting such programs to UNIFLEX as long as they are written in Pascal, C, FORTRAN or another portable language.
Torstein Krogh, PROGRAM 67, Postboks 926, 8010 BODIN,Norway

*****

TRS 80 **COLOR COMPUTER** TRS 80

SPECTRAL ASSOCIATES, the LEADER in TRS-80 Color Computer software, presents the best in 16K machine language, fast action, high resolution arcade games. Extended Basic not required.

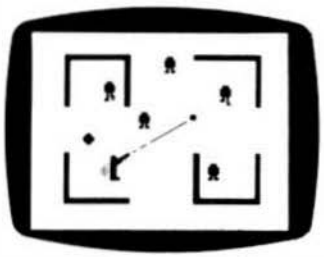


COSMIC INVADERS

The best invaders type game available. Complete with 16 skill levels, dynamite sound and 4 color hi-res graphics. Can you dodge the bombs long enough to blast the alien ship and invaders? $21.95

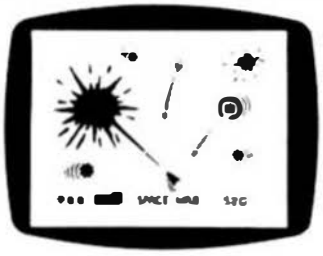
GHOST GOBBLER

An exciting version of the popular PAC-MAN game with power dots, 8 bonus shapes, super sound and 16 skill levels. Hours and hours of challenging fun. Joysticks $21.95



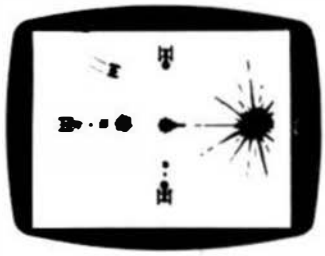
ROBOT BATTLE

Can you dodge the never ending horde of robots while avoiding the fatal touch of the Android? Realistic voices and 16 skill levels provide a tremendous arcade type challenge. Joysticks, $21.95.



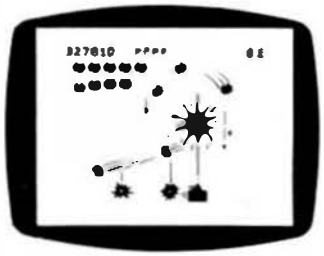
SPACE WAR

Maneuver your spaceship through the defenses of the Death Star for the killing shot while avoiding space mines, enemy ships and the gravity pull of the BLACK HOLE. Two color hi-res graphics - $21.95 Joysticks



COLOR ZAP

Super new space game! You must dodge the incoming photon torpedos before they can penetrate the defenses of your ship. Once you think you've got the upper hand, defensive fighters may appear and attack. $9.95



GALAX ATTAX

If you like Space Invaders, you'll love Galax Attax! Alien fighters leave formation to attack your ground base and you must fight them off! Exciting, fast-action machine language game with super sound and high resolution graphics. 16K and joysticks required. $21.95

NEW 16K MACHINE LANGUAGE GAMES

- KEYS OF THE WIZARD $19.95
  Best new adventure game available - over 200 rooms filled with creatures, tricks, treasures, magic spells. Cassette save feature built in.
- COLOR OUT $9.95
  Like Breakout, you must knock out six layers of blocks using your paddle to prevent missiles from escaping. Joysticks optional.

SUPERCHARGER

If you have a 32K E series printed circuit board, the **SUPERCHARGER** will let you switch your computer to full 64K RAM mode. It just plugs into the ROM PAK port and you can use BASIC or not as you wish. NO MODIFICATION is needed and it will not void your warranty. It allows you to still plug ROM pack and/or the disk controller board into the computer. ONLY $39.95

THE FACTS

At last, a complete description of the "guts" of the Color Computer. Specs on all the ICs, complete schematics, theory of operation and programming examples.

$14.95

FLEXPLUS DOS

FLEXPLUS is a powerful, easy-to-use disk operating system. Spectral Associates has adapted TSC's FLEX to the best DOS completely compatible with Radio Shack software for use on the Color Computer. Eliminate the need for Radio Shack's TRS DOS - use FLEXPLUS with Editor/Assembler and have the options of a full range of utilities. FLEXPLUS works on the 32K Radio Shack disk system with 64K memory chips with a High Resolution multi screen format that supports a 24 line by 51 character display! Also included are special enhancements to Radio Shack's Disk system when you are running FLEX with single or double sided, single or double density, 35, 40 and 80 track drives.

Advantages of FLEXPLUS are:

- Best price anywhere
- Easy start-up — just type "RUN FLEX+"
- Allows you to save RS compatible disk files from **FLEXPLUS**
- All FLEX compatible software will run including INTERRUPT DRIVEN SOFTWARE
- NO HARDWARE MODIFICATIONS NEEDED
- Warranty will not be voided — no need to open computer
- Wide range of available software
- Requires Supercharger board
- $149.95 for **FLEXPLUS**, Editor/Assembler and Supercharger

UTILITIES

- EDITOR/ASSEMBLER $34.95
- SUPER MONITOR 19.95
- EPROM PROGRAMMER 89.95
  (Program your own ROMs for the ROM PAC port)
- MAGIC BOX 24.95
  Load MOD I/III Basic Tapes into color computer

SPECTRAL ASSOCIATES
139 Harvard Ave.
Tacoma, Washington 98466
(206) 565-8483

WRITE FOR COMPLETE CATALOG

ADD 3% FOR SHIPPING NO COD

VISA OR MASTERCHARGE ACCEPTED

All prices in U.S. funds

ORDERS ONLY TOLL FREE

800-426-1830

EXCEPT WA, HI, AK

TEN MOST-ASKED QUESTIONS

ABOUT DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. **What is an electronic spread-sheet, anyway?**
   Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. **Is DYNACALC just for accountants, then?**
   Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. There's even a built-in sort command, so you could use DYNACALC to manage small data bases - up to 256 records.

3. **What will DYNACALC do for ME?**
   That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications.

4. **Do I have to learn computer programming?**
   NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Built-in HELP messages are provided for quick reference to operating instructions.

5. **Do I have to modify my system to use DYNACALC?**
   Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. **Will DYNACALC read my existing data files?**
   You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, RMS data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. **How fast is DYNACALC?**
   Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. **Is there a version of DYNACALC for MY system?**
   Probably. You need a 6809 computer (32k minimum) with FLEX or UniFLEX operating system. A version for OS-9 is also in the works. You also need a decent CRT terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX version of DYNACALC also allows you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX and Data-Comp's F-MATE. A version for Frank Hogg's Color Computer FLEX is also being done.

9. **How much does DYNACALC cost?**
   The FLEX versions are just $200 per copy; UniFLEX version $395. Foreign orders add $10 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. **Where do I order DYNACALC?**
    See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 a.m. to 6 p.m., Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX versions. Software serial number is required for the UniFLEX version of DYNACALC.

ORDER YOUR DYNACALC™ TODAY

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics. 7A Burton St., Darlinghurst. NSW 2010 Sydney. Telephone: 02-357-5111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

CSC

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

ALSO FROM CSC

DYNAMITE +
"THE CODE BUSTER"

now available for UniFLEX
OS-9 version soon

DYNAMITE + is a new version of DYNAMITE, our popular 6809/6800 disassembler package for 6809 FLEX. Present users of DYNAMITE can upgrade to DYNAMITE + by sending us the original DYNAMITE diskette and $40 (plus $5 for foreign postage). DYNAMITE + does everything DYNAMITE does, and more! A cross-reference generator has been added, label files are now maintained only in text form (LABEL EQU $xxxx), and boundary file specifications have been tremendously simplified, which makes it easier to disassemble large programs containing lots of big tables.

The UniFLEX version of DYNAMITE + does everything the FLEX version does, and also automatically handles system calls and 'info' areas.

DYNAMITE + is available for $100 per copy on FLEX (specify diskette size), and $300 on UniFLEX. Foreign orders add $5 per copy for postage.

UniFLEX software prices include maintenance for the first year.

DYNACALC, DYNAMITE, and DYNAMITE + are trademarks of Computer Systems Center.

F-MATE is a trademark of Data-Comp.
VisiCalc is a trademark of VisiCorp.
OS-9 is a trademark of Microware and Motorola.
FLEX and UniFLEX are trademarks of TSC.

OmegaSoft Puts Pascal Power In Program Development

Finally. A fast, efficient way to design, code, debug and maintain your microprocessor software. OmegaSoft's Pascal. The software development tool that takes the sting out of programming. For industrial or business use, large or small scale.

How does it work? Simple. We've compiled Pascal into the compact native code that is most common to professional applications. OmegaSoft Pascal is designed to integrate easily with existing software development tools available on most 6809 operating systems.

Single Pass Punch

Our compact single-pass compiler quickly translates Pascal into 6809 assembly language code. This means less time-wasting disk swapping and fewer multiple passes, with greater success rates. OmegaSoft Pascal is based on the proposed ISO standard with compatible extensions designed to make Pascal an important tool in industry and business. Byte wide variables allow convenient access to I/O devices. Custom I/O devices can be used in place of normal Pascal INPUT and OUTPUT files.

Practical Yet Precise

OmegaSoft's Pascal long, 32-bit integer support also allows representation of money amounts without speed penalties of BCD implementations. And our 7 digit real format is compatible with AM9511 APU with options to use this chip instead of software routines.

OmegaSoft Pascal's random access file is excellent for fast access of data for business applications programs and other mass storage uses. Sophisticated applications programs are simplified with dynamic length strings, such as:

- Process control.
- Communication systems.
- Robotics.
- Automatic test systems.
- Microcode generation.
- Scientific computation.
- Development system software.

High Speed, High Performance

OmegaSoft Pascal generates high speed code without sacrificing ROMability, position independence or reentrancy. Modular compilation is supported along with several methods of interfacing to assembly language variables and routines, including interrupt procedures.

OmegaSoft Pascal's runtime library provides that only the modules necessary to execute the Pascal program are loaded, which translates into much smaller object modules and reduced memory costs.

Our symbolic debugger interfaces with the assembly language output of the compiler for fast, accurate program execution. Debugger commands allow other exciting possibilities:

- Displaying and changing variables using Pascal names.
- Tracing through statements.
- Setting breakpoints at the start of Pascal statements.

Features

- Compatible with MDOS™ and XDOS™, FLEX™, DOS69™, and OS-9™.
- Entry, External, and Absolute attributes for variables and procedures.
- Optimized in-line code generated wherever possible rather than subroutine calls.
- Compiler generates 6809 assembly language – can either be used with the debugger or assembled and linked to form a loadable object module.
- Standard data types plus long-integer, hex, dynamic length string, and device.
- Expanded operations for pointers and characters allows low level code to be written in Pascal.
- Extensive type conversion functions.
- Compiler – Requires 48K system.
- Symbolic Debugger to run on any of the operating systems.
- Utilities to assist in creating a "CHAIN" file for automatically linking the user program with the runtime library.
- Manuals – detailed language reference handbook and a configuration manual.
- Runtime library and source.
- ISO validation report available.
- Debugger allows display and change of variables using their symbolic names and stepping through the program one statement at a time.

Single unit prices start at $425 (domestic).

Support products include: OmegaSoft's Motorola compatible Relocatable Assembler/Linker and for higher speed OmegaSoft's Arithmetic Processor option uses the AMD9511 to support integer, longinteger, and real calculations for Pascal runtime. Available soon will be a multitasking kernel and a high speed translator between compiler output and relocatable object code.

TM: MDOS, XDOS, Motorola trademarks; Flex, TSC trademark; DOS69, Smoke Signal Broadcasting trademark; OS-9 Microware trademark.

OmegaSoft
P.O. Box 70265
Sunnyvale, CA 94086
Telephone (408) 733-6979

SCREEN EDITOR/WORD PROCESSOR FOR DOS6x, FLEX, OS-9

IMPORTANT INFO

THE BEST NEWS!

ALFORD & ASSOCIATES
P.O. BOX 6683
RICHMOND, VA., 23230
804-320-6722

CoCo SPEECH SYNTHESIZER

TRANSLATOR FOR SP1

SSB UTILITIES DISK

FANTASTIC PROOFREADER

SSSO SPEECH SYNTHESIZER

TRANSLATOR FOR VS1

TREK-68

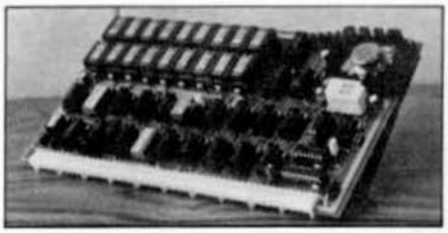
COLOR-TREK

WINCHESTER FOR MOTOROLA EXORCISOR/MDOS

☐ 10 MB Winchester hard disk runs MDOS on Motorola Exorcisor System ☐ No modification to MDOS required ☐ MDOS based software stays alive ☐ All user software operates without modification ☐ Optional SA-801R flexible diskette drive system ☐ Optional 10 MB removable cartridge.

CSA *For information call (714) 566-3911*
Computer System Associates
7562 Trade Street, San Diego, CA 92121

QMM1
Quarter Megabyte Memory



The **QMM1** is a "Quarter Megabyte" (256K) dynamic parity memory board for SS50-C 6809 systems with 20 bit addressing.

- Full 2Mhz operation with transparent on board refresh, runs continuously at 2Mhz with no cycle stretching, stealing or halting.
- Optional parity feature halts processor and sounds audible alarm upon detecting a read error.
- Switch disable of up to 8 2K, 4K, or 8K blocks in any or all banks.
- Gold bus connectors, socketed, Professional quality board.
- On board transient surge and ESD protection.
- All boards assembled, burned in, tested and warranteed for 1 year.

NOTES: Not for 6800 systems. Requires 1 jumper addition to 6809 CPU board.

	config. #	without parity	config. #	with parity
64K	110	$ 495.00	221	$ 550.00
128K	130	$ 695.00	241	$ 775.00
192K	350	$ 935.00	461	$1045.00
256K	370	$1135.00	481	$1270.00

ORDER FROM:
D.P. Johnson
7655 SW Cedar crest St.
Portland, OR 97223
(503) 244-6152
Delivery:
2-4 weeks ARO

TERMS: Cash, Check, or Krugerands with order, add $5.00 shipping and ins. in USA, foreign orders include shipping for 1.5 kilos.

OS9 Application Software
Specialty Electronics, Inc.

ACCOUNTS PAYABLE

Take the "headache" out of tracking your accounts payable with Specialty Electronics interactive Accounting system. The accounts payable package supports these outstanding features:

1. Entry of debits credits regular invoices full and partial payments
2. Hand checks entered directly or computer printed checks with stubs and check ledger
3. Provides aging of accounts and proper general ledger distribution of all entries
4. Job costing, customer order number tracking and buyer classification are provided
5. Vendors may be added as needed
6. Complete audit trails are provided
7. Reports can be generated for specific due dates, customer groups, open or closed items and in either voucher detail or vendor summary formats

Accounts Payable I-code
$299

GENERAL LEDGER with CASH JOURNAL

The general ledger is the center of the Specialty Electronics Accounting System. With the package you can:

1. Produce balance sheets and income statements in various formats
2. Define account names, looping, positioning, headings and subaccounts
3. Format special reports and print percentages
4. Post by hand, cash journal or by using the interactive accounts receivable, payable and payroll
5. Provide a clear audit trail for all entries
6. Input data in an easy to follow format
7. Use for multi company accounting without modification

General ledger I-code
$399

ACCOUNTS RECEIVABLE

Your Accounts Receivable can be followed with a minimum of time investment using these features:

1. Regular invoicing, debit and credit memos, full and partial payments
2. Progressive billing and payment
3. Aging of periods specified by the user
4. New customers entered as needed
5. Statements are generated listing individual invoices and overdue amounts totaled by aging category
6. Total interaction with the general ledger with tax, property and travel expenses compiled separately and posted to various accounts

Accounts Receivable I-code
$299

INVENTORY

The Specialty Electronics interactive accounting System Inventory Control Package provides the tools for complete control of a large and active inventory providing:

1. Reports for quantities on hand, quantities on order, activity and many other categories
2. Complete item classification, category groups, supplier information, order dates, reorder quantities, etc.
3. Simple input and recount option procedures

Inventory Control I-code
$299

PAYROLL

The Specialty Electronics interactive Accounting System provides payroll support which goes beyond writing paychecks. Its features include:

1. Weekly, bi-weekly, semi-monthly and monthly pay periods
2. Hourly, salary, vacation, holiday, commission, overtime and compensatory pay categories and rates
3. Deducts federal and state payroll taxes, insurance, additional tax or special deductions
4. Daily time keeping allowed
5. Prints checks, stubs, check ledger and journal history
6. Prints W-2 forms, federal and state tax report information
7. Keeps full employee history
8. Tax tables allow user modification

Payroll I-code
$425

Complete Documentation $19.95

* OS9 and Basic 09 are trademarks of Microware, Inc. and Motorola Corp.

P.O. Box 541
2110 W. Willow

Specialty Electronics

(405) 233-1632
Enid, OK 73701

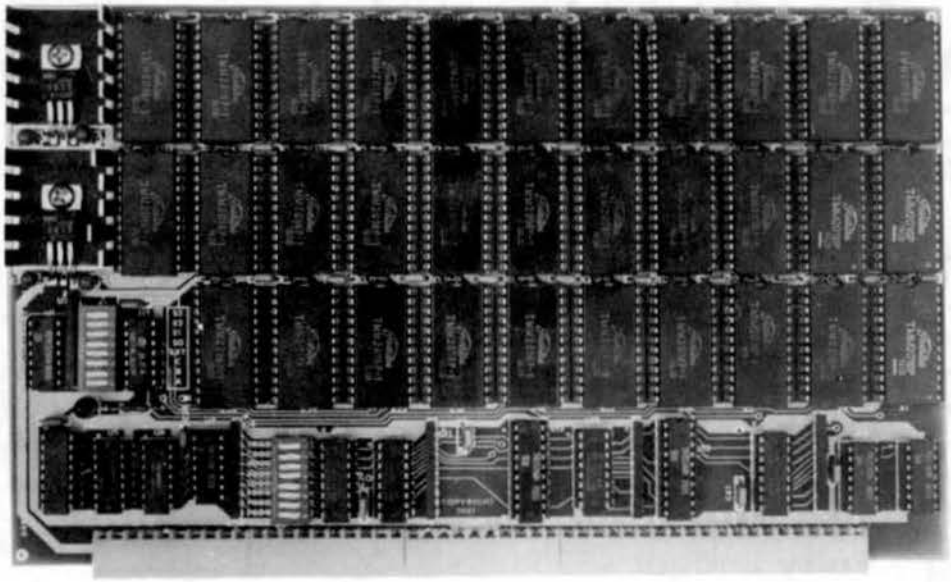
64K SS-50 STATIC RAM

NEW!

$259.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $40**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K Kit	$309
64K Kit	$359

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

AAA Chicago Computer Center

ELEKTRA CABINET Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 8-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives.

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 18v, 3a @ -16v. 3 primary inputs for light, rated, and heavy loading.

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives. Heavy duty version for 1 Winchester drive and 1 floppy drive.

ELEKTRA MOTHERBOARD Heavyweight 0.125" thick. 18" long by 9" wide. 11 memory (50 pin) slots. 4 or 8 slots may be cut off for shortening to 14" or 10" lengths respectively. 8 I/O (30 pin) slots. Complete address decoding of I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. Base address for I/O slots can be placed at 32, 64, or 128 byte increments respectively. 1" spacing between all memory and I/O slots. Extended addressing capability for both memory and I/O ports for meeting SS-50C bus specifications. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1MHz 30 pin disk controllers to run with 2 MHz 50 pin CPU boards.

ELECTRA CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2716 Eproms, 1K scratchpad, MC6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. The board supports DMA by either HALT or BUSREQ when a 6809 CPU is used. The board does not support a DAT and therefore does not support extended addressing. The board will run any of the MIKBUG™ compatible monitors in the 6802/6808 mode and SBUG-E, HUMBUG, and GMXBUG-09 in the 6809 mode. The ELEKTRA CPU 8/9 will run any of the popular disk controller boards with the appropriate software. OS-9™ Level 1 is available.

ELEKTRA DPS DUAL PORT SERIAL CARD Fits 1 standard 30 pin SS-50 bus I/O slot. Can be configured for 4 addresses per port with the B port 2 addresses higher than the A port or for 16 addresses per port with the B port 4 addresses higher than the A port. Each port is terminated at two 18 pin dip sockets, one socket configured for modem and the other for terminal or printer. RTS, CTS, DTR, DCD, and DTR are appropriately implemented. Each port has independent selection of baud rate. Each port allows the interrupt request to be independently jumpered to the IRQ or FIRQ/NMI bus line.

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be used in either the 4 or 16 addresses per I/O slot configuration occupying the first four addresses of the I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request lines for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, assembled motherboard with gold square pin connectors $825.00*

ELEKTRA CABINET	260.00
ELEKTRA 110v POWER SUPPLY	175.00*
ELEKTRA CABINET, 110v POWER SUPPLY, AND POWER SUPPLY CABLES	410.00*
ELEKTRA CABINET, 110v PS, PS CABLES, STD. DISK REGULATOR AND CABLES	460.00*
*Add $30.00 for 220v	
STANDARD DISK REGULATOR BOARD AND CABLES	50.00
HEAVY DUTY DISK REGULATOR BOARD AND CABLES	75.00
FILLER PLATE FOR 5-1/4" DRIVE OPENING	10.00
FAN FILTER	10.00
GOLD 10 PIN CONNECTORS (Specify male w/square pins or female)	1.50
TIN 10 PIN CONNECTORS (Specify male w/square pins or female)	.50

	BARE BOARD	KIT	ASSEMBLED
ELEKTRA MOTHERBOARD (Gold connectors)	80.00	320.00	380.00
ELEKTRA MOTHERBOARD (Tin connectors)		240.00	300.00
ELEKTRA CPU 8/9	50.00	225.00	275.00
ELEKTRA CPU 8/9 with baud rate option		250.00	300.00
ELEKTRA DPP DUAL PORT PARALLEL BOARD**	20.00	60.00	80.00
ELEKTRA DPS DUAL PORT SERIAL BOARD**	20.00	60.00	80.00
**CABLE FOR DPP OR DPS (2 needed; specify board)			20.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drives, and software along with I/O terminals to form working computer systems.

AAA CHICAGO COMPUTER CENTER

120 CHESTNUT LANE • WHEELING, IL 60090

(312) 459-0450

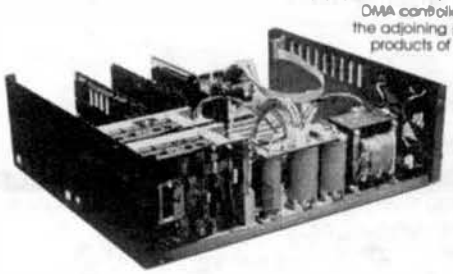
Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

ELEKTRA

COMPUTER PRODUCTS



The CPU, 56K memory board, and DMA controller board in the adjoining picture are products of GIMIX, Inc.

SMOOTH™ Software

SUPER MODEM PROGRAM

Transmit manually to distant computer.

Transmit disk files (text) of any length to distant computer.

Receive and save disk files (text) of any length on local disk system. If sending computer does not support an X-on/X-off protocol, then the received files are limited in size by the computer memory.

Tested to transmit and receive text at speeds up to 9600 baud. (CRT terminal must be capable of operating at a baud rate higher than the one the modem is operated at.)

Half duplex option in case distant computer doesn't echo.

Echo option so user can simulate a time sharing system. (Super Modem Program doesn't support auto-answer but the source is provided for those individuals who wish to adapt our program to their special needs.)

Replaces CR with CR/LF (user option) for those using time sharing systems that don't transmit LF's.

Slow disk file transmit (user option) based on character verification for use on time sharing systems to which disk files cannot be sent at speed suggested by the baud rate.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".

Manual and disk with both source and object code $75.00

STANDARD MODEM PROGRAM

Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Reception of disk files is limited to those small enough to completely fit within the receiving buffer.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".

Manual with instructions, source listing, and flow chart; disk with both source and object code $45.00

Manual with instructions, source listing, and flow chart 25.00

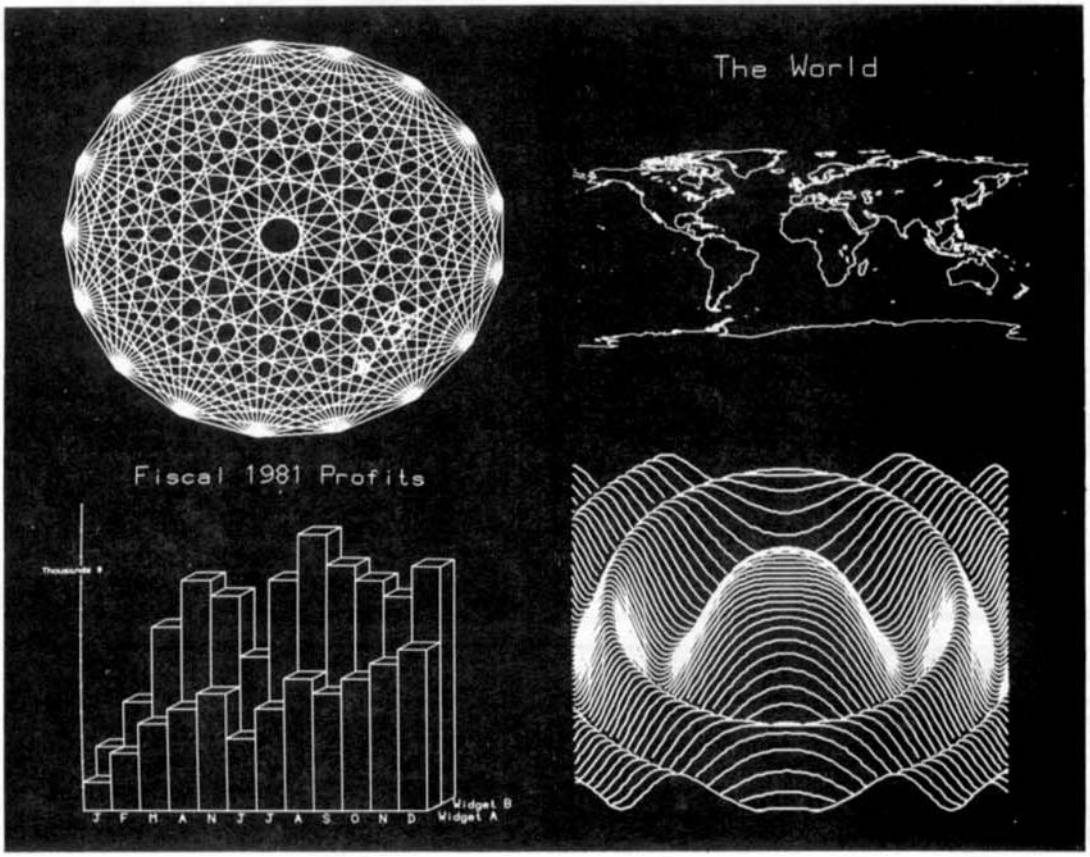
Modems (U.S. Robotics)	Manual Answer	Auto Answer
1200 Baud (120 cps), direct connect	449.00	549.00
300 Baud (30 cps), direct connect	179.00	219.00
300 Baud (30 cps), acoustic		149.00

ANNOUNCING ElectroScreen™

only $595

the Superior Alternative

to the Traditional Alphanumeric Terminals



The ElectroScreen™ Intelligent Graphics Board Features:

Graphics

- 512 x 480 resolution bit-mapped display
- Interleaved memory access — fast, snow-free updates

Intelligence

- 6809 on-board mpu
- 6K on-board firmware
- STD syntax high level graphics command set
- Removes host graphics software burden
- Flexible text and graphics integration
- Multiple character sizes
- User programs can be run on-board

Terminal

- Terminal emulation on power-up
- 83 characters by 48 lines display
- Easy switching among user-defined character sets
- Fast hardware scrolling

Additional Features

- SS-50C and SS-64 compatible board
- Board communicates with host through parallel latches
- Composite and TTL level video output
- 8 channel 8 bit A/D converter
- Board occupies 4 address bytes

See your dealer today!

The ElectroScreen manual is available for $10. credited toward purchase of the board.

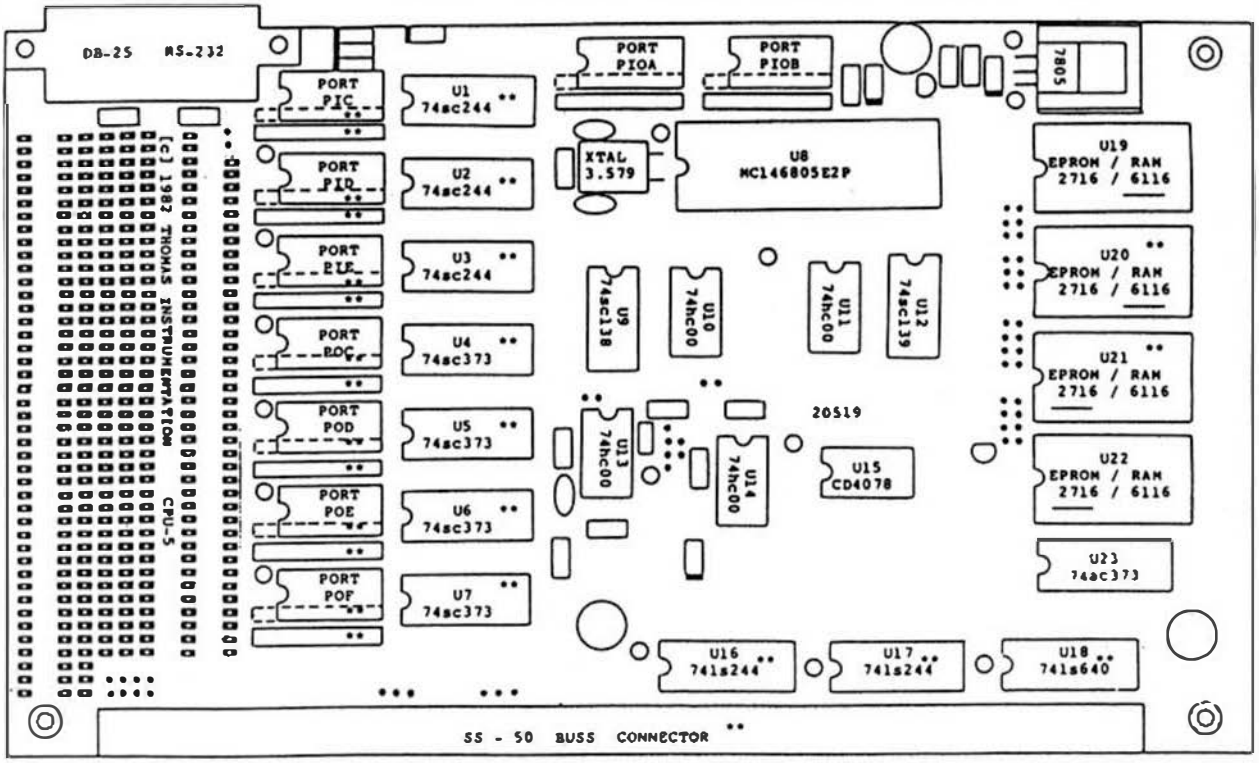
The ElectroScreen has a 90 day warranty from purchase date.

Dealers, please contact us for our special introductory package.

Privac Inc (703) 671-3900

3711 S. George Mason Dr., Falls Church, Va. 22041

*** NEW 6805 ALL CMOS CARD ***



CPU-5-B	Blank card with documentation	$49.00	
CPU-5-P	Partial card (parts marked ** not incl.)	$229.00	Includes EPROM Monitor
CPU-5-F	Full card with all parts shown (fully assembled)	$349.00	and 115 vac Power Pak.
CROSS-ASSEMBLERS	for FLEX (c) TSC 6809 system	$150.00	

SOFTWARE

VDISK Treat extended memory as a super fast disk drive
6809 source & object $149.00
6809 object $ 99.00

OUTSIDE MODEM PROGRAM incl. source
UniFLEX version $100.00
FLEX version 6800 & 6809 $ 50.00
HAYES SMART MODEM $249.00

CROSS ASSEMBLERS for 6800,6801,6805 runs on 6809 FLEX $150.00

TV-EDIT Screen oriented Editor for FLEX 6809 $ 95.00
Please specify 5" or 8" Disk when placing order.

CHIPS

Please note when ordering IC's that a $100.00 Minimum is in effect Orders less than the minimum will be charged an additional $10.00 for handling

6116P-3	2MHZ	$13.50
2016P-2	2MHZ	$10.50
2716	1MHZ	$ 6.50
2716-1	2MHZ	$16.50
146805E2P		$25.00
68B02		$10.00
68B09		$25.00
68B21		$ 6.50
68B40		$12.00
68B50		$ 6.75
74LS640		$ 3.25

HARDWARE

NEW MEMORY PRICES TO INTRODUCE VDISK

S-R/R without memory chips	$120.00
with 8K NMOS@2MHZ	$169.00
with 16K NMOS@2MHZ	$199.00
with 32K NMOS@2MHZ	$299.00
with 48K NMOS@2MHZ	$399.00
Bare Card	$ 49.00

Extender Cards assemb. with logic aid

SS-50/50C	$ 35.00
SS-30	$ 25.00
SP-1 Prototype Board A/T	$195.00
SP-1 Bare Card	$ 49.00
SS-50 Wire Wrap Board B/C	$ 39.00
SS-30 Wire Wrap Board B/C	$ 20.00
6802 Super CPU A/T	$235.00
6802 Super CPU B/C	$ 59.00
Video RAM, B/C $49.00, A/T	$195.00
Parallel I/O, B/C $49.00, A/T	$139.00
SS-50 Backplanes 4,6,8,12,614 position @ $3.00 per slot w/o connectors	
SS-30 8 pos. BP w/o connect.	$ 39.00
Transition Card $49.00, A/T	$ 95.00
Molex Gold Male $1.60, Female	$ 1.60
Molex Tin Male $ .40, Female	$ .50

THOMAS INSTRUMENTATION

168 EIGHTH STREET — AVALON, N.J. 08202 (609) 967-4280
NJ RES. INCLUDE 5% SALES TAX
Allow 3 Weeks for Checks to Clear
CONT. USA INCLUDE $3.00 SHIPPING, CANADA $6.00, FOREIGN $12.00
MASTERCARD, VISA, and C.O.D. ACCEPTED
Send $20.00 to receive full documentation, schematics, & source listings for all boards currently in production

'68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

THE CHIEFTAIN™ 5¼-INCH WINCHESTER HARD DISK COMPUTER

SO ADVANCED IN SO MANY WAYS . . . AND SO COST-EFFECTIVE . . . IT OBSOLETES MOST OTHER SYSTEMS AVAILABLE TODAY AT ANY PRICE.

● HARD DISK SYSTEM CAPACITY

The Chieftain series includes 5¼- and 8-inch Winchesters that range from 4- to 60-megabyte capacity, and higher as technology advances. All hard disk Chieftains include 64-k memory with two serial ports and DDS69D disk operating system.

● LIGHTNING ACCESS TIME

Average access time for 5¼-inch Winchesters is 70 msec, comparable to far more costly hard disk systems. That means data transfer ***ten-times faster*** than floppy disk systems.

● 2-MHZ OPERATION

All Chieftains operate at 2-MHz, regardless of disk storage type or operating system used. Compare this to other hard disk systems, no matter ***how*** much they cost!

● DMA DATA TRANSFER

DMA data transfer to-and-from tape and disk is provided for optimum speed. A special design technique eliminates the necessity of halting the processor to wait for data which normally transfers at a slower speed, determined by the rotational velocity of the disk.

● RUNS UNDER DOS OR OS-9

No matter which Chieftain you select . . . 5¼- or 8-inch floppy, or 5¼- or 8-inch Winchester with tape or floppy back-up . . . they ***all*** run under DOS or OS-9 with ***no need*** to modify hardware or software.

● UNBOUNDED FLEXIBILITY

You'll probably never use it, but any Chieftain hard disk system can drive up to 20 other Winchesters, and four tape drives, with a single DMA interface board!

● SMOKE SIGNAL'S HERITAGE OF EXCELLENCE

This new-generation computer is accompanied by the same ***Endurance-Certified*** quality dealers and end-users all over the world have come to expect from Smoke Signal. And support, software selection and extremely competitive pricing are very much a part of that enviable reputation.

20-Megabyte Tape Streamer Back-Up Option

Available with all Chieftain hard disk configurations. This cartridge tape capability provides full 20-megabyte disk back-up in less than five minutes with just one command, or copy command for individual file transfers. Transfers data tape-to-disk or disk-to-tape. Floppy back-up is also available in a variety of configurations.

The Chieftain Computer Systems:

Here are the Chieftain 6809-based hard disk computers that are destined to change the data processing industry . . .

☐ **CHIEFTAIN 95W4**
4-megabyte, 5¼-inch Winchester with a 360-k floppy disk drive (pictured).

☐ **CHIEFTAIN 95XW4**
4-megabyte, 5¼-inch Winchester with a 750-k octo-density floppy disk drive.

☐ **CHIEFTAIN 98W15**
15-megabyte, 5¼-inch Winchester with a 1-megabyte 8-inch floppy disk drive.

☐ **CHIEFTAIN 9W15T20**
15-megabyte, 5¼-inch Winchester with a 20-megabyte tape streamer.

Write or call today for details (including the remarkably low prices) on the total Chieftain Series . . . and on dealership opportunities.

SMOKE SIGNAL BROADCASTING®
31336 VIA COLINAS
WESTLAKE VILLAGE, CA 91362
TEL (213) 889-9340

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Telephone ( ) _____

2MHZ 6809 SYSTEMS

GIMIX offers you a variety to choose from!

38 MB WINCHESTER SYSTEM $17,498.99

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ 512KB Static RAM
- ★ 8 RS232C Serial Ports
- ★ 2 Parallel Ports
- ★ DMA Double Density Floppy Disk Controller
- ★ Dual 8" DSDD Floppy Disk System
- ★ Dual Winchester Subsystem with Two19 MB 5¼" Winchester Drives

SOFTWARE FEATURES:

- ★ OS-9 LEVEL TWO Multi-User Operating System
- ★ OS-9 Debugger
- ★ OS-9 Text Editor
- ★ OS-9 Assembler

19 MB WINCHESTER SYSTEM $8998.09

HARDWARE FEATURES:

- ★ 128K Static Ram
- ★ 2MHz 6809 CPU
- ★ 19 MB 5¼" Winchester DMA Subsystem
- ★ 4 RS232C Serial Ports
- ★ 1 MB 5¼" Floppy Disk Drive
- ★ DMA Double Density Floppy Disk Controller

SOFTWARE FEATURES:

- ★ OS-9 LEVEL TWO Multi-User Operating System
- ★ OS-9 Text Editor
- ★ OS-9 Debugger
- ★ OS-9 Assembler

128KB MULTI-USER SYSTEM $6997.39

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ DMA Double Density Floppy Disk Controller
- ★ 128KB Static Ram
- ★ 2 RS232C Serial Ports
- ★ Dual 8" DSDD Floppy Disk System

SOFTWARE FEATURES: Your choice of either UniFLEX or OS-9 LEVEL TWO. Both are Unix-like Multi-User/Multi-Tasking Operating Systems.

56KB FLEX / OS-9 "SWITCHING" SYSTEM $4148.49

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ 56K Static Ram
- ★ 2 RS232C Serial Ports
- ★ DMA Double Density Floppy Disk Controller
- ★ 2 Built-in 5¼" 40tr DSDD Disk Drives (80 Track DSDD Drive Option . . add $400.00)

SOFTWARE FEATURES:

- ★ GMXBUG monitor — FLEX Disk Operating System
- ★ OS-9 LEVEL ONE Multi-tasking operating system for up to 56K of memory

WINCHESTER SUBSYSTEMS

Winchester packages are available for upgrading current **GIMIX** 6809 systems equipped with DMA controllers, at least one floppy disk drive, and running FLEX, OS-9 LEVEL ONE or OS-9 LEVEL TWO. The packages include one or two 19MB (unformatted) Winchester drives, DMA Hard Disk Interface, and the appropriate software drivers. The Interface can handle two 5¼" Winchester Drives, providing Automatic Data Error Detection and Correction: up to 22 bit burst error detection and 11 bit burst error correction. **UniFLEX NOW AVAILABLE**

Dual drives can be used together to provide over 30 MBytes of on line storage – or use one for back-up of the other. (More convenient and reliable than tape backup systems.

#90 includes one 19MB Drive, Interface, and Software **$4288.90**
#91 includes two 19MB Drives, Interface, and Software **$6688.91**

Contact GIMIX for systems customized to your needs or for more information.

50 HZ Export Versions Available

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

GIMIX INC.

FLEX & RS COLOR COMPUTER

If you are tired of playing games on your TRS-80C™ Color Computer, or find that you are handicapped by the limitations of the RS BASIC in trying to write a Program that will allow you to actually USE the Color Computer as a COMPUTER; and if you have been studying the Advertisements in this Magazine and wishing that you could run THESE Programs on your Computer, YOU ARE READY TO MOVE UP TO THE FLEX9™ Operating System. If you want to have REAL PROGRAMMING POWER, using an Extremely Powerful Business BASIC, PASCALs, C Compilers, a full-blown Macro Assembler with a Library capability so you are not continuously "reinventing the wheel", YOU ARE READY TO MOVE UP TO THE FLEX9™ Operating System. If you would like to see if YOU REALLY COULD USE A COMPUTER IN YOUR BUSINESS, or begin to make your Computer start PAYING IT'S OWN WAY by doing some Computer Work for the millions of small businesses around you, such as Wordprocessing, Payroll, Accounting, Inventory, etc., then YOU ARE READY TO MOVE UP TO THE FLEX9™ Operating System. How?? DATA-COMP has the way!

DATA-COMP's FLEX9™ Conversion for the TRS-80C™ Color Computer was designed for the SERIOUS COMPUTER USER: with features like greatly increased Display Screens, WITH Lower Case Letters, so you can put a FULL Menu on ONE Screen, or see SEVERAL Paragraphs at the same time; with features like providing a FULL Keyboard so you have FULL Control of your Computer AND it's Programs NATURALLY, without needing a chart to see what Key Combination will give you what function; with USER ORIENTED functions to make using the Operating System natural, like having the Computer AUTOMATICALLY determine what type of Disk is being used in what type of Disk Drive and working accordingly, rather that you have to specify each and every thing for it, or like having the Computer work with the Printer you have been using all along without you having to tell the new Operating System what is there; etc., etc., etc.

DATA-COMP has everything you need to make your TRS-80C™ Color Computer WORK for YOU; from Parts and Pieces to Full, Ready To Use SYSTEMS. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

-- SYSTEM REQUIREMENTS --

FLEX9 Special General Version w/Editor & Assembler (which normally sell for $50.00 ea.)	$150.00
F-MATE(R) FLEX9 Conversion Rout. for the RS Disk Controller when purchased with Special General FLEX9 Sys.	$49.95
when purchased without the General FLEX9 Sys.	$59.95
NEW -- Full Source Code for the Conv. Routines	$159.95
Set of Eight 64K RAM Chips w/ Mod. Instructions	$99.95
Color Computer with 64K RAM and EXT. BASIC	$549.95
Color Computer with 16K RAM	$375.95
Color Computer with 16K RAM and EXT. BASIC	$465.95

-- SPECIAL SYSTEM PACKAGES --

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9™, F-MATE(RS)™, and a Box of 10 Double Density Diskettes; a COMPLETE, ready to run SYSTEM on your Color TV Set.
$1249.95

-- FOR USERS THAT ALREADY HAVE FLEX9 & Disk Drives --

Radio Shack DISK CONTROLLER with F-MATE(RS)™ and a Special Two Drive DISK CABLE $254.95

-- DISK DRIVE PACKAGES, with RS Controller --

These Packages Include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable:

PAK #1 ==> 1 Single Sided, Double Density Sys.	$499.95
PAK #2 ==> 2 Single Sided, Double Density Sys.	$729.95
PAK #3 ==> 1 Double Sided, Double Density Sys.	$579.95
PAK #4 ==> 2 Double Sided, Double Density Sys.	$889.95

-- PARTS AND PIECES --

Radio Shack Disk Controller	$169.95
1 ea. Single Sided, Double Density Disk Drive	$249.95
1 ea. Double Sided, Double Density Disk Drive	$349.95
Single Drive Cabinet with Power Supply	$79.95
Double Drive Cabinet with Power Supply	$99.95
Single Drive Disk Cable for RS Controller	$24.95
Double Drive Disk Cable for RS Controller	$34.95
Micro Tech. Prods., Inc. LOWER CASE ROM Adepter	$74.95
Radio Shack BASIC Version 1.1 ROM	$34.95

SOFTWARE

CHESS 6809

Requires FLEX™ and one of the following CRT terminals

Now Runs On Any Type Terminal

Features:

- Two display boards.
- Four levels of play.
- Point scoring system.
- Play white or black.
- Change or set-up boards/piece positions.
- Forfeit move.
- Swap sides.
- Make move and swap sides.
- Change skill level.
- Stop and restart game.
- Solve 'Mate in 1-2-3-4' moves.

$79.95 Specify 5" or 8" disk

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600 +.

Note: Personal checks allow 3-4 week delivery.

DIET-TRAC Forecaster

A Diet Planning and Analysis Program

DIET-TRAC Forecaster is a program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C.P.F. %) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individuals are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's predicted weight for a 30-day period is printed.

FLEX VERSION	$59.95
UniFLEX VERSION	$89.95

PRINTERS

The Epson MX-80
$495.00
The Epson MX-100
$725.00

MX-70 $355.00 MX-80 FT $575.00

MEMORY

SWTPC-Motorola, MP32
32K Dynamic Memory Board
Assembled & Tested
1 MHZ - No extended addressing
Can be set up for $0-7FFF or 8000-FFFF

$149.95

MasterCard VISA

DATA-COMP SOUTH EAST MEDIA

P.O. Box 794 *HIXSON, TN 37343*
1-615-842-4601

Verbatim Diskettes.

5" Soft Sector Disks

Single Side Single Density	$2.75 ea.
Single Side Double Density	$2.75 ea.
Double Side Double Density	$4.92 ea.
Plastic Storage Box	$2.00 ea.

8" Soft Sector Disks

Single Side Single Density	$3.75 ea.
Single Side Double Density	$4.10 ea.
Double Side Double Density	$4.75 ea.
Plastic Library Box	$5.00 ea.

Foreign Orders Add 10% Surface—20% Air Mail

DRIVES & CABINETS W/PS

5 1/4"

TANDON - Single Sided, Double Density, 40 Track.....$249.95

TANDON - Double Sided, Double Density, 40 Track.....$349.95

CABINET - Single Drive with Power Supply.....$ 79.95

CABINET - Double Drive with Power Supply.....$ 99.95

CABLE - Single Drive.....$ 24.95

CABLE - Double Drive.....$ 34.95

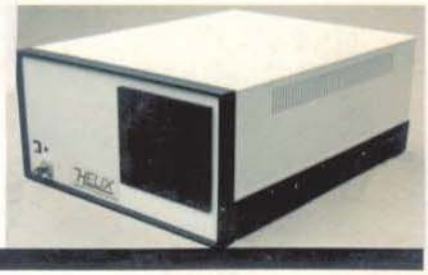
* NOTE - When ordering cables please specify S50 Bus or Other!!!

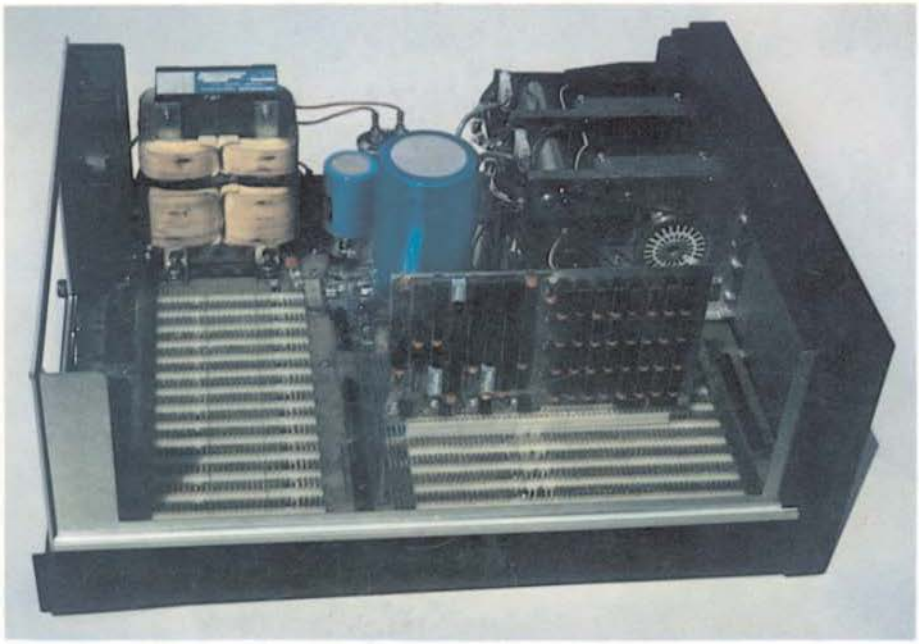
Call or write for disk controller Board Information.

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

HELIX™





THE MAINFRAME

- Industry Standard Optima™ Cabinet
- Largest Constant Voltage Power Supply in the industry
- S-64 Bus gives 16 Bit Power and S-50 Bus Compatibility
- 10 Main (S-64) Slots
- 14 I/O (S-30) Slots plus 2 On-board
- On-board Baud Rate Generator to 38.4Kb
- Space and Power for two 5¼" Disk Drives
- Full Address Decoding for I/O Slots
- Two RS-232 Serial and Two parallel Ports On-board
- Single Board Construction for Reliability
- Faraday Shielded Bus Lines give "Text Book Clean" Signals

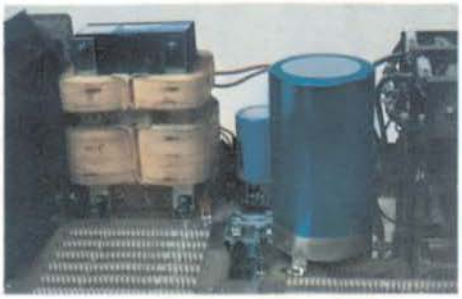
THE PROCESSORS

6809

- Standard 2 MHz Operation
- Standard DAT Compatible with GIMIX and SWTPC
- Standard 6840 Interval Timer
- Standard 1K Scratchpad RAM
- Standard Clock/Calendar with Battery
- Provision for Programmers Console

68000

- Standard 8 MHz Operation
- Memory Management Hardware
- Provision for Programmers Console
- 16 Bit Power and 8 Bit Compatibility



The HELIX™ computer system represents the latest advance in S-50 bus computer systems. Relying on the physical nature of S-50 bus connectors to guarantee compatibility, the HELIX adds 14 bus lines (becoming S-64) to allow a 68000 processor to operate with full 16 bit data transfer and 24 bit addressing, while at the same time providing full interchangability with existing S-50 components.

Offered with a selection of processors, memories, and peripheral controllers, a HELIX system can be configured for applications ranging from advanced hobbyist to multiterminal time-sharing.

Designed to offer the utmost in speed, reliability, and utility at a reasonable price, it represents a new standard of quality for those who require a professionally designed computer for professional use.

THE POWER SUPPLY

- Ferro-resonant Transformer for Line Noise and Under-Voltage Protection
- Conservative 25 Amps at 8.5 Volts
- Conservative 5 Amps at ±16 Volts
- Conservative Component Rating for Reliability

THE COMPONENTS

- Fully Socketed
- Gold Plated Bus Connectors
- Only "B" Series 68XX Components Used
- Only Top Grade Logic Circuits Used
- Industrial Grade Components Throughout

THE MEMORIES

DM-64

- Field Proven
- Proprietary Memory Control Logic
- Fully Transparent Refresh
- Tested at 2.5 MHz Operation

DM-512

- 512K Bytes on a Single S-64 Board
- 16 Bit Power and 8 Bit Compatibility
- Runs in Existing S-50 Systems where Physical Space Allows
- Full 24 Bit Addressing
- Fully Transparent Refresh

THE PRICES

Because of the variety of configurations possible, full pricing cannot be given. Representative prices are:

- 64K 6809 HELIX $1995
- 64K 68000 HELIX $2595
- 512K 6809 HELIX $4450
- 512K 68000 HELIX $4995

HAZELWOOD COMPUTER SYSTEMS

907 E. Terra, O'Fallon, Missouri 63366 (314) 281-1055

Dealer and OEM Inquiries Invited. We support our Dealers.

Optima is a Trademark of Scientific-Atlanta